SUN ネットワーク詳説

相田　仁
小池汎平 共訳

Analyzing SUN Networks

Carl Malamud

Includes
TCP/IP
and
NFS

丸善株式会社

本書は，TCP/IP上で開発された，ネットワークファイルシステム(NFS)や遠隔手続き呼出し(RPC)をはじめとするSunのONC(Open Network Computing)プロトコルについて詳しく解説したものです．

さらにネットワーク構築にあたっての重要な話題であるネットワーク管理，セキュリティ，相互運用性等についても詳述されており，コンピュータネットワークに関心のあるすべての方々にとって必須の知識を提供しています．

SUN ネットワーク詳説

相田　仁
小池汎平 共訳

Analyzing SUN Networks

Carl Malamud

Includes TCP/IP and NFS

丸善株式会社

ANALYZING SUN NETWORKS

by

Carl Malamud

はじめに

サン・マイクロシステムズ(Sun Microsystems)は，常に開放型システムの概念の上に立ってビジネスを行ってきた．当初のSunワークステーションは市販の部品で作られていた．標準的な部品を用いて最新技術のワークステーションとソフトウェアを作り上げることは今日まで継続されている方針である．

この哲学はネットワークにも及んでいる．Sunのネットワークは，国際的なInternetネットワークで用いられているのと同じ，業界標準のTCP/IPプロトコルの上に作られている．この本は，不均質な，高速かつ大規模なネットワークを作り上げるのに，TCP/IPのネットワークをいかに用いることができるかについて示したものである．

TCP/IPはネットワークの出発点でしかない．この本ではまた，ネットワークファイルシステム，RPC，その他Open Network Computing(ONC)ファミリに属するいくつかのプロトコルについても説明する．主要なすべてのオペレーティングシステムの，100万台以上のコンピュータで，広くONCが受け入れられていることを考えれば，この本は単にSunのネットワークというよりも，TCP/IPとNFSを用いるすべてのコンピュータに関する本といえるであろう．

業界標準に加えて，ネットワーク管理，セキュリティ，相互運用性，その他ネットワークの管理に重要ないくつかの話題も含まれている．いかにしてOSIやSNIをネットワーク中に作成し，いかにしてその不均質なコンピューティング環境を管理するかについて述べる．

実現可能なネットワークアーキテクチャを作ることがこの本の主題である．本文中を通じて，Sun自身のネットワークおよびその他の先進的な場所が事例研究として用いられる．この本を書いている間の悦びの1つは，実際に存在し，世界中で用いられている技術に取り組むことであった．

この本がAnalyzing Networks 3 部作の最後の巻である．OSIに基づくネットワークに関しては，読者はAnalyzing DECnet/OSI Phase Vを参照されたい．また，独自のネットワークに関しては，Analyzing Novell Networksを参照されたい．今日運用されている不均質なネットワークに関しては，ページをめくられたい．

Carl Malamud / carl@malamud.com

謝辞

Sun MicrosystemsのネットワーキングおよびデータSD通信部長のGeoffrey Baehrには，Analyzing Sun Networkに関する本というアイデアを現実にすることを助けていただいた．大きなコンピュータ会社との過去の何度かの出会いの後，私はSun Microsystemsが優れた技術や，開放的な，独自でないシステムを指向していることを知って悦びを感じた．Terri Davisからは，とても書ききれないほどいろいろな面で助けていただいた．

以下に挙げるSun Microsystemsの技術者たちからは，原稿の査読と，数多くの質問に対する答えをいただいた: Sally Ahnger, Karl Auerbach, Gigi Babcock, Teresa Barr Beyer, Gabriele Cressman-Hirl, Linda Cwiak, Frank DeMarco, Jonathan Feiber, Dennis Freeman, Burt Fujii, Terence Gibson, Eric Johnson, Tom Kessler, Abhijit Khale, David Kipping, Fred Lowe, Karen Maleski, Milton Mallory, Mike Martinez, Dennis McLain, Chuck McManis, Bill Melohn, Bob Plummer, Ian Pope, Vipin Samar, Warren Smith, Warren Smith, Mark Stein, Richard Thio, Ian Vessey, Keith White, Yinpo Wong, そしてDennis Yaro.

AuspexのBruce Nelson，ISIのJoyce K. Reynolds，NASA-Ames研究センターのWilliam T.C. KramerとLouise D. Kokinakis，サンディエゴスーパーコンピュータセンターのJay Dombrowski，Trusted Information SystemsのSteve CrockerとHilarie K. Orman，UnidataのRobert Green，そしてデラウェア大学のDavid Mills博士からはいずれも貴重なご助力をいただいた．

Network GeneralのHarry Saal博士からは，プロトコルを身近にするためのSniffer Network Analyzerを再度貸していただいたほか，原稿に対する貴重な査読をいただいた．

Sun MicrosystemsのBrian Pawlowskiには，原稿全体の詳細な技術的査読をしていただいた．彼の細部にわたる注意と，技術レベルの高さが，最終版に大きく寄与している．私は彼から，彼が「真に温かく面白いやつである」と書き加えるよう頼まれた．そうしようと思う．

Van Nostrand ReinholdのDianne Littwinはこのシリーズ３冊を通じての編集者であった．彼女の大変な仕事と信じられないほどの忍耐に対して，心から感謝の意を表したい．彼女は休暇に値する．

謝辞

Sun Microsystemsのネットワーク・システムエンジニア部部長のGeoffrey Baehrには、Analyzing Sun Networksに関するあらゆるアイデアを具体化することを支援していただいた。大きなコンピュータ会社における製品開発の間の一つとして、私はSun Microsystemsが優れた技術を、開発技術を学ぶことを支援していることを知った。Terri Dennisからは、とても素晴らしいはかりしれない支援をいただいた。

以下に挙げるSun Microsystemsの技術者たちからは、原稿の査読と、技術上の質問に対する答えをいただいた：Sally Ahnger, Karl Auerbach, Olga Babcock, Teresa Burr Beyer, Gabriele Cressman-Hirl, Linda Cerruti, Frank DeMarco, Jonathan Fisher, Dennis Foreman, Bud Fong, Terence Gibson, Eric Johnson, Tom Kessler, Ashim Khan, David Kopping, Fred Lowe, Karen Malaski, Milton Mallory, Mike Martinez, Dennis McLain, Chuck McManis, Bill Melohn, Bob Plummer, Ian Pope, Vipin Samar, Warren Smith, Warren Smith, Mark Stein, Richard Thio, Bill Vassey, Keith White, Yuepo Wong, そしてDennis Yau。

AmpexのErnst Nelson、IBMのJoyce K. Reynolds、NASA Ames研究センターのWilliam T. C. KramerとLouise D. Rokimakos、オーストラリアのテレコムのJay Dombrowski、Trusted Information SystemsのSteve CrockerとHilarie K. Orman、UnilabのRobert Crepp、そしてカリフォルニア大学のDavid Millsからは、いずれも貴重なご意見をいただいた。

Network GeneralのHarry Saalからは、アドバイスと資料をいただき、Sniffer Network Analyzerを利用させていただいた。原稿に対する貴重な意見をいただいた。

Sun MicrosystemsのBrian Pawlowskiには、原稿全体の査読と技術的な助言をしていただいた。彼の経験にもとづく意見で、皆様もこれを助けられたであろうと思う。

Van Nostrand ReinholdのDianne Littwinとのスケジュールの調整には大変お世話になった。

訳者まえがき

この本はCarl Malamud著の“Analyzing SUN Networks”の訳である．表題には“SUNネットワーク”とあるものの，原著のまえがきにもあるとおり，この本の内容はSUNワークステーションに限られるものではない．その理由の第1は，この本で取り上げられているNFSやRPC，NISなどの，いわゆるOpen Network Computing(ONC)技術は，SUNが他のワークステーションメーカにライセンス公開をしたために，DEC，HPをはじめとするほとんどすべてのワークステーションに採用されていることである．また，理由の第2は，この本の内容が，SUNワークステーションに現在採用されている技術にとどまっていないことである．例えば第13章では電子メールに関してSunOSをはじめとするバークレー版UNIXの標準であるsendmailシステムと，ISO標準のX.400システムの両方が述べられている．また，第8章で述べられているNTPや第15章の管理などは，多数のワークステーションを持つサイトにおいて重要となってきている技術であるが，SUNワークステーションに標準としてバンドルされているものではない．

このように，この本はSUNワークステーションのネットワークに限らず，現在の世界のコンピュータネットワークの中心であるTCP/IPプロトコルに基づくネットワーク(いわゆるInternet)すべてに適用される基本技術について述べたものと言うことができるであろう．

また，この本の内容は原則としてSunOSのバージョン4.0に基づいて書かれているが，最新版のOSであるバージョン5(Solaris 2)においても，ファイル名などが一部異なるものの，共通に適用される内容であると考えてよい．特に第9章で述べられているNIS+はSolaris 2になってはじめて標準としてバンドルされるようになったものである．

本書の翻訳に当たっては，東京大学工学部の齊藤・相田研究室と，田中英彦・小池研究室が合同で輪講を行い，そのときに用いた原稿を元に相田と小池が分担して最終稿を作成した．分担は次のとおりである．

まえがき他，奇数章，用語集	相田
偶数章	小池

輪講および原稿作成に当たって参加・協力してくれた両研究室の方々にこの場を借りて感謝の意

を表したい.

原文は，あまり接続詞などを用いずに話題がどんどん切り替わるという，意味の取りにくい文章であり，内容の誤りも散見される．翻訳に当たって，ある程度接続詞を補ったり，訳注を付けたりしたが，残念ながら充分満足できるレベルには必ずしも至っていない．読者の方々からのご指摘，ご叱責を乞う次第である.

最後に，原稿の進みの遅い訳者達を辛抱強く待ち続けて下さり，また用語の統一などに大変な尽力をして下さった丸善出版事業部の方々に，心から感謝の意を表したい.

1993年8月

訳者代表　相田　仁

商標に関する注意書き

PostScriptはAdobe Systemsの商標である.

Apple, AppleTalk, LocalTalk, およびMacintoshはApple Computerの商標である.

dbaseはAshton-Tateの商標である.

IngresはASK, Inc.の商標である.

STREAMS, Transport Layer Interface, Unix, およびUnix System V Release 4はAT&Tの商標である.

AuspexはAuspexの商標である.

CRAYはCray Research, Inc.の商標である.

CullinetおよびIDMSはCullinetの商標である.

DEC, DECconnect, DECmcc, DECnet, DNA, Message Router, VAXcluster, およびVMSはDigital Equipment Corporationの商標である.

EpochおよびInfiniteStorageはEpochの商標である.

AS/400, DISOSS, IBM, IBM PC LAN, PC/AT, PC/XT, PROFS, SNA, SNADS, System/370, System/38, および3270 Display StationはInternational Business Machines, Inc.の商標である.

MS-DOSおよびMicrosoftはMicrosoftの商標である.

Network GeneralおよびSniffer AnalyzerはNetwork General Corporationの商標である.

HYPERchannelはNetwork Systems Corporationの商標である.

Network File System, NFS, Open Network Computing, ONC, SPARC, Sun SunOS, およびTOPSはSun Microsystems, Inc.の商標である.

SybaseはSybaseの商標である.

Carl MalamudはCarl Malamudの商標である.

目次

Chapter 1

序論

1.1 概要

ある人達にとってはネットワークはフリーウェイと同じようなものである．それはそこにあり使わなければならないので，1つか2つの出口を覚えて，毎日の通勤のために行ったり来たりを繰り返す．もう少し詮索好きな人達にとって，高速道路はそれぞれ異なるサービスを提供するいろいろな種類の出口をもったバックボーンのようなものである．

この本は，このような詮索好きな人達に対して道路地図を提供しようとするものである．この本では，伝送制御プロトコルおよびInternetプロトコル(Transmission Control Protocol and Internet Protocol : TCP/IP)に基づくネットワークがどのように動作するかに関して，読者が何らかの関心をもっているものと想定している．

TCP/IPは単にSunワークステーションというよりもずっと広範囲に及んでいる．したがってこの本の表題は必ずしも適当ではない．この本の最初の部分はTCP/IPを実装しているどんなコンピュータにもあてはまるものであり，それは現在動作している真空管を使わないすべてのコンピュータを意味する．

この本では，いろいろなネットワークを均質な一体に統合するためにTCP/IPをいかに用いることができるかを見ていく．個々のネットワークはFDDI，イーサネット，X.25，ISDNというようにいろいろと異なる回線で構成することができる．これらのいろいろな種類のデータリンクに関しては第2章で詳しく述べる（FDDIといった頭字語(acronym)の簡単な説明に関しては適宜用語集を参照せよ）．次いで第3章ではTCPやIP（あるいはその他のいくつかのプロトコル）がこれらのサブネットワークの上にどのように構築され，それらをインターネットワークへと統合するかを示す．

TCP/IPの上に，Sun Microsystemsで当初開発されたいろいろなサービスが形づくられている．それらはオープンネットワークコンピューティング(Open Network Computing : ONC)という公式な名前，あるいは離れたコンピュータ上のファイルシステムを直結のドライブと同様にみせる

ことのできるプロトコルであるネットワークファイルシステム(Network File System : NFS)というあだ名の下に，一群としてまとめられている．NFSは広範囲な分散処理，特にディスクをもたない(diskless)ワークステーションに対して門戸を開いている．

しかしNFSはONCのごく一部にしかすぎない．ONCの提供する遠隔手続き呼出し(remote procedure call)プロトコルを用いて，他の数々のサービスが構築されている．例えばネットワーク情報サービス(Network Information Service : NIS)は名前サービスである．ロック管理(lock manager)やその他いろいろな支援サービス(support service)もある．SunNet Managerや3270カラー端末エミュレーションなどのような他のアプリケーションもまた，このONCインフラストラクチャを用いている．

ONCはこの本の多くの部分における主題である．第5章と第7章ではRPCプロトコルとXDRプロトコルについて議論する．第6章ではNFSについて述べる．その他のONCアプリケーションについても，自動マウンタ(automounter)について第11章で，資格サーバ(lisence server)とロック管理(lock manager)については第12章で述べる．

ONCやTCP/IPプロトコルについて述べたとしても，この本が“Sunの”ネットワークに関する本であることを本当の意味で限定するものではない．確かにNFSを開発したのはSunであるが，開放的なライセンス方針のため，NFSは主要なオペレーティングシステムのすべてに溶け込んでいった．

したがって，この本の真の内容は非均質な(heterogeneous)ネットワークの構築法をSunワークステーションを例にとって述べたものである．ここで述べる技術の大部分は広い範囲のオペレーティングシステムやハードウェアに適用することが可能である．

この本の主眼は，Sunをはじめとするサーバやワークステーションからなる非均質なネットワークがどのようにして動作するかを理解することにおかれている．いろいろな種類の話題について，できる限り，最もわかりやすいような順序で詳しく述べる．話題の選び方はもちろん任意である．必ずしもすべての問題点に触れたわけではないし，どの問題点に関しても絶対的な方法で扱おうとしてもいない．より進んだ文献の紹介が，ソフトウェアを作成したり，システムをチューンしたり，その他ネットワークを管理するための実践作業を行う人のために用意してある．

しかし，たとえ問題が起こったときにはマニュアルの後を追いかけざるを得ないとしても，かなり詳しい概要説明というものは役に立つであろう．どんな部品が含まれているかわかっていれば，リファレンスマニュアルやプロトコル仕様書に記載された詳細に対する貴重な見通しが得られるというものである．この本全体を通じて，ネットワークの構成(configuration)，ネットワークアナライザの画面ダンプ，システムコールの一覧表，参照表といった図や表が現実に対する感覚をもたらしている．ある読者はこの情報を“生データ”と名づけていた．筆者はこの言葉を良い意味として解釈している．すなわち，理論を上手に表現する具体例であると．

これらの生データは与えられたプロトコルやモジュールの能力に対する理解を助けるものとし

て読者に示されている．ネットワークアナライザのダンプをみることはエラーの状況をデバッグするために有用であるばかりでなく，パケットがどのように構成されているかを理解するための学習手段としても有用である．どのようなシステムコールが利用可能であるか，あるいはモジュールの中にどのような管理情報が格納されているかということを知ることもまた同様に教育的である．もちろん，これらの生データはすべて注釈という健全な薬によって補足されている．図ばかりの本は素敵ではあろうが，ネットワーク解析の絵本は別の機会にゆずることとしよう．

Sun 広域ネットワーク(SWAN)は本書の全体を通してケーススタディとして用いられる．Sun から提供されたケーススタディ資料に加えて，サンディエゴスーパーコンピュータセンター，NASA-Ames および Internet からの資料も引用している．Network General の Sniffer Analyzer からの一連の画面ダンプがケーススタディを補足している．この装置は多少の変更を加えた IBM PC で，トークンリングやイーサネット，ARCnet，その他のネットワークに接続して，データを選択して捕捉することができる．

そしてその後，収集したデータを解析する．Sniffer Analyzer を用いたのは，収集した情報を表示する形式が，技術的内容としては驚くほど普通の英語に近いからである．

一部の読者にとっては必要とするすべてのもの，すなわち TCP/IP や NFS その他のプロトコルの環境でネットワークがどのようにして動作するかに関する概念的な全体像が得られるであろう．しかし，大部分の読者は Sun ワークステーションや TCP/IP ネットワークとおそらく何らかの関わりをもつようになるであろう．さもなければこの本を読まないであろうから．そのような読者にとっては，この本はマニュアルや機能仕様，ハードウェア図面などの生データやその他のコンピュータに付随するすべてのデータに対する補足となる．Sun の良い点の 1 つは，このようなデータの多くが壁一面を占領する文書の形で出荷されるのではなく，電子的に得られるようになった点である．

この本は全体像を提供するものである．もし読者がネットワークの利用者であるならば，この本の提供する全体像は，遠隔ファイルシステムを NFS でマウントしたときに何が起こっているかや，なぜライセンスサーバはアプリケーションを先に進めさせてくれないのかといったことを理解する手助けとなるであろう．ネットワーク上で利用するためのアプリケーションを書くことに興味をもっている人達もいるかもしれない．どのようなツールが利用可能であるかを理解して適切な選択ができるようにすることは重要である．例えばデータベースサーバの位置を登録するのに NIS を用いるというように，既存のサービスを利用することで，プログラマが 1 から開発する必要がなくなる．

利用者とソフトウェアプログラマとはこのような情報に有用性を見出す 2 つの人種であるが，それ以外にもう 1 つの人種がある．それはネットワークを購入，設置(install)，管理しなければならない人達である．この種の読者にとっては，ある内容は現在の専門レベルと比べて初歩的であるかも知れない．FDDI チップを設計する技術者は FDDI(やイーサネット)に関してこの本から

PID	TT	STAT	TIME	COMMAND
0	?	D	0:00	swapper
1	?	IW	0:00	/sbin/init -s
2	?	D	0:00	pagedaemon
55	?	S	1:02	portmap
58	?	IW	0:00	ypbind
60	?	IW	0:00	keyserv
68	?	S	18:51	in.routed
71	?	I	0:00	(biod)
72	?	I	0:00	(biod)
73	?	I	0:00	(biod)
74	?	I	0:00	(biod)
84	?	IW	0:01	syslogd
92	?	IW	0:02	/usr/lib/sendmail -bd -q1h
96	?	IW	0:00	rpc.statd
98	?	IW	0:00	rpc.lockd
105	?	S	0:00	automount
112	?	S	10:46	update
115	?	IW	0:00	cron
120	?	IW	0:16	inetd
123	?	IW	0:00	/usr/lib/lpd
253	?	IW	0:00	rpc.rquotad
383	co	S	0:04	-csh (csh)
519	co	R	0:00	ps -ax

1-1 典型的なワークステーション上のプロセス

学ぶことは何もないであろうが，メッセージシステムに関してならば得るものがあるかもしれない．

この本は第一の試みである．市場調査の白書や貿易新聞よりはずっと内容が豊富であるが，もともとの出典文献すべてよりはもちろん少ない．人間の性質が変わらないことを望みつつ，数センチの厚みの本は数メートルに及ぶ一連の出典文献よりも魅力的であるという事実を，筆者はよりどころとしているのである．

図 1-1 は多数の画面ダンプのうちの最初のものである．このダンプは Sun ワークステーションからのものであり，コンピュータ上で現在存在するすべてのプロセスを表示したものである．このリスト中の 11 の異なるプロセスについてこの本で述べる．この本を読み終わった後で，実際のネットワークに関与しているのはどのプロセスで，それらのプロセスが何をしているかを明らかにすることは，読者に対する（したがってまた著者に対する）面白いテストである．

Chapter 2

サブネットワーク

2.1　はじめに

おおかたの期待通り，最初は基本的なものから始めることにする．2台のコンピュータを結ぶ1本の線(wire)からである．この線は，2台以上のコンピュータがつながっていてもよく，高速なものでも低速なものでもよく，安価なものでも(通常はこちらだが)高価なものでもよい．サブネットワークと呼ぶことにするこのようなサービスの例の1つとして，線で接続された任意の2つのノード間でデータの交換を行うことがあげられる．

ここでいう“線”という用語はもちろん抽象的なものである．本書で述べるようなサービスを提供することは，2つのモデムと電話回線を用いてもできるし，イーサネット，X.25，FDDI，トークンリング，ISDNなど，現在利用可能などのようなサブネットワーク技術を用いてもよい．

多くのネットワークアーキテクチャと同様，TCP/IPの世界もまた上で述べたサービスが限界に達するところから始まる．イーサネットはTCP/IP階層の底にあたるIPプロトコルから使用される．これは，多くの他のネットワークアーキテクチャ，すなわちOSI，Novell社のNetWare，DECnet，そして(やり方に違いはあるが)IBM社のSystem Network Architecture(SNA)などにおいても同じである．

用語について一言いっておこう．サブネットワークという言葉は，アーキテクチャが異なるたびに，様々な異なった言外の意味をもつ．これはルータ，ゲートウェイ，ブリッジなどの用語についても同じである．この本では，サブネットワークという言葉は，あるホストが中間ノードのサービスを利用せずに他のホストへ直接データを送ることができるとみなせる，任意のシステムのことを意味するものとする．あるホストから別のサブネットワークへデータを送るときには，ルータと呼ばれる中間ノードのサービスを利用する．ブリッジというのは，T1専用回線を介して2つのLANを接続するイーサネット間ブリッジ(Ethernet-to-Ethernet bridge)などの例のように，サブネットワーク内で通信範囲を拡大する装置である．より精密な用語はMAC層ブリッジであり，ブリッジの処理がデータリンク層の媒体アクセス制御副層(medium access control sublayer)

にあてはまることを意味している．

サブネットワークという用語は少々不正確である．多くの場合，この用語は，具体的にイーサネットか FDDI LAN のことを意味している．しかし，この言葉の本来の意味は，ローカルに接続されているとみなせる任意のノードの集合のことである．ノード間には，IP 層でパケットの中継を行う一連のルータがあってもよい．この章では，サブネットワークについて，イーサネットや FDDI のようなデータリンク層技術の観点から調べる．IP をベースとしたホストが，ルーティングの決断を行うときに，多くの異なるサブネットワークの集まりを 1 つの大きなサブネットワークであるかのように扱えることは，次の章で示される．

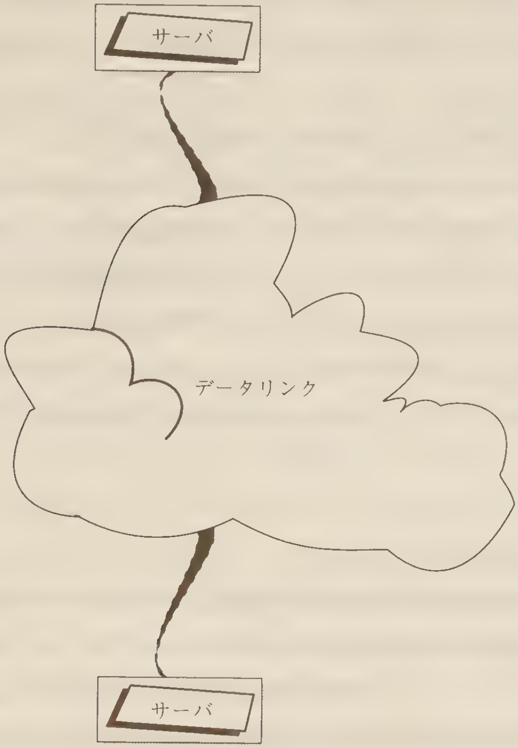
現在利用可能なサブネットワーク技術の大部分では IP プロトコルが使用されており，これらのサブネットワーク技術について，アマチュア無線から FDDI やギガビット LAN や WAN システムにいたるまで見ていくことにする．ただし，この調査の目的は，徹底的に網羅した調査をすることではなく，単に利用可能な例を示すことである．

この章では，イーサネットや FDDI などのいくつかの基本的なサブネットワーク技術について述べ，次にいくつかの LAN や WAN の環境においてこのようなサブネットワークの組合せがどのように組織されてきたかについての一連のケーススタディを行う．このケーススタディも網羅的なものではなく，いくつかの組織において実際にどのようにしてサブネットワークを組み合わせることを決断したかを示すだけにするつもりである．重要な考え方は，この本の残りの部分で述べられるすべてのサービスが，基本的にはこれらのすべてのサブネットワークで均一に機能するということなのである．

ネットワークが高速になると処理も高速に実行されるが，NFS を用いてデータのブロックを書くことなど，処理の意味内容は変わらない．低速で雑音の多い電話回線で，回線切断が起きた場合のようなエラー発生時には動作に違いが生じることもあろうが，処理がうまくいっている限りは同じように機能する．

特定のアプリケーションを，それを引き受ける低位層に適合させることは難しいかもしれない．多くの場合，組織は単に下位のインフラストラクチャを作り，利用の総計を観測するだけである．場合によっては，特定のアプリケーションが専用のサブネットワークに移されることもある．多くの場合，NFS サーバとクライアントはイーサネットを共有し(近接していることがこの場合は良い性能をもたらす)，2 台の Cray コンピュータは 1Gbps の Ultranet ネットワークを共有して独自のクロスマウントを行う．

サブネットワークでの IP の基本的な考え方は，ある方向へあてられたすべてのトラフィックを決められた一連のゲートウェイに向けることができるということである．これらのゲートウェイはアドレスをより詳しく調べ，実際にどのようにしてパケットを誘導するかを選択するだろう．しかし，これらのすべての技術における基本的なサービスは，サブネットワークに接続された任意の 2 つのノードの間でデータのパケットを移動させることである．この意味で，すべてのサブネ



2-1 雲

ットワーク技術は雲のようなものと見なすことができる(図2-1参照). なぜなら,上位層が依存するのは基本的なサービスだけだからである. データリンク層と物理層の役割は, パケットを移動させるために必要なすべての複雑さを, サービスの利用者から切り離すことである.

この本を通して,データリンクサービスの様々な利用者を見ることになる. TCP/IPネットワークにおける主な利用者はInternetプロトコル(IP)ネットワーク層である. しかし,このネットワーク層以外にも, アドレス解決プロトコル(Address Resolution Protocol)などの他のプロトコルや, DECnetやOSIなどの他のネットワーク層もある.

ネットワークはこれまで特定のコンピュータどうしを結びつけるデータリンクだけからなることが多かった. このようなポイント間プロトコルは, 多くの場合, 広域環境での動作を意味していた. これはコンピュータが高価で,各サイトにコンピュータが1台しかなかった時代のことである. このようなポイント間プロトコルの1つの例は, IBMの2進同期(Bisynch)プロトコルである. DECの計算機環境においては,初期のPDPとVAXシステムはディジタルデータ通信メッセージプロトコル(Digital Data Communication Message Protocol : DDCMP)を利用していた. TCP/IPの世界では, シリアル回線IP(Serial Line IP : SLIP)とポイント間プロトコル(Point-to-Point Protocol : PPP)が専用回線上で使用されている.

ローカルな環境において, ポイント間プロトコルはイーサネットプロトコルなどの多重アクセスLANに急速にとって代わられた. Sunネットワークの中心はイーサネットである. すべての

Sun コンピュータには組込みのイーサネットアダプタがついてくるし，端末サーバやプリンタサーバなどのサードパーティシステムはすべてイーサネットに適合するように設計されている．

より高速な LAN 技術を利用するために，イーサネット以外にも多くのシステムが構築されている．HYPERchannel や Proteon は 50 ないし 100Mbps の範囲で動作するシステムとしてよく用いられているが，これらのシステムの多くは FDDI ネットワークにとって代わられようとしている．これらの高速データリンクは，異なるサブネットワークを接続するバックボーンとして用いられたり，スーパーコンピュータから得られたシミュレーション結果をグラフィック視覚化する場合など，高速なデータ転送が要求されるアプリケーションで用いられる．

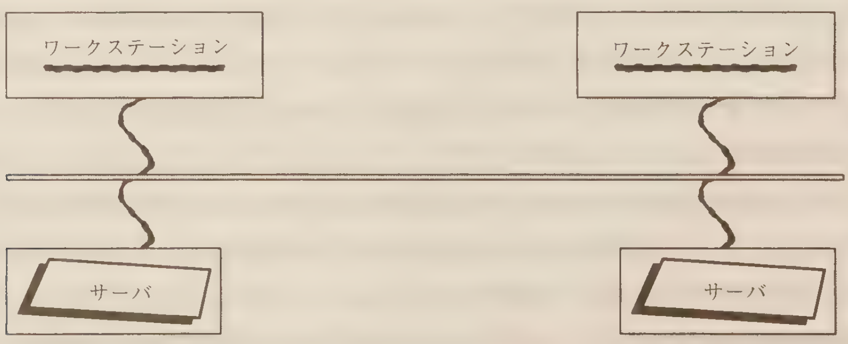
広域環境では，ポイント間接続は，X.25 やフレームリレー，SMDS，ISDN など新たに出現してきた技術に基づくパケット交換ネットワークと組み合わせられて用いられている．

2.2　イーサネット

イーサネットは当初 Xerox の Palo Alto 研究所で開発され，その後 IEEE において 802.3LAN プロトコルとして標準化された．潔癖主義者のために一言ことわっておくと，この本ではイーサネットという用語を元来のイーサネットと，その後の 802.3 標準の両方に用いる．それら 2 つの間を区別する必要があるときには IEEE 標準を 802.3 と呼び，元来の DEC/Xerox/Intel 仕様の最新版をイーサネットレベル 2 と呼ぶ．

いかなるイーサネットにおいても，基本的な構成は共有バスである(図 2-2 参照)．利用者から見れば，イーサネットは多数のノードの接続された一本の電線である．どのノードも他の任意のノードに向かってパケットを送ることができる．

すぐ後で述べるように，単一のバスあるいは電線という論理的な描像は，現実の物理的構成に比べて簡単化のしすぎである．リピータと呼ばれる装置を用いることで多数の異なるセグメントを接線して，マルチセグメントのイーサネットとすることができる．このようなマルチセグメントのイーサネットは，今度はブリッジを用いて接続して拡張イーサネットとすることができる．



2-2　バス

このように組み合わされた電線やコネクタは，利用者からは論理的にはバスに見える．すなわち任意のノードから任意のノードにパケットを送ることができるのである．

2.2.1 CSMA/CDプロトコル

イーサネットの基本的動作は，CSMA/CDというキャッチフレーズで呼ばれているプロトコルによって行われている．CSMA/CDはキャリアセンス-多重アクセス／衝突検出を意味している．キャリアセンスとは，論理バスを共有しているすべてのノードは，他のノードが送信中かどうか検知することができることを意味している．もし送信中であるなら，正常動作のイーサネットノードは待つ．多重アクセスとは，もし媒体が使用中でないならどのノードも特別な許可を待つことなく送信することができることを意味している．

任意の2つのノードは送信することができるので，2つのノードが聴取(listen)して媒体が空いていることを検知し，同時にデータを送出することがあり得る．その結果，2人の人が同時に話すのと同様な衝突(collision)がネットワーク上で起こる．人間は衝突を無視するかもしれないが，イーサネットでは両方のノードが衝突を検出して送信を中止する．両方のノードはその後乱数時間だけ（それら2つのノードにおいて時間は異なる）待ってから媒体が空いているかどうかを調べる．もし空いているならば送信を開始してよい．

何も送信できないまま衝突を繰り返すことが理論的にはあり得るが，現実には衝突が繰り返される可能性は極めて低い．代表的かつ典型的なイーサネットの利用率は10～15%であり，すべてのノードが送信しようとしたときにごく少ない衝突で送信することが可能である．しかし，過負荷のネットワークにおいて平均の利用率が50%を越えることはそれほどめずらしいことではない．その場合には性能低下のつけが回ってくる．

もし，あるネットワークのすべてのノードが単一のイーサネットに接続されていたとすると，ネットワークを飽和させることがあり得る．しかし，現実的には，多くのネットワークは複数のイーサネットに接続される．その上でブリッジまたはルータを用いて複数の部署(work group)を接続し，単一の統合化された拡張LANとするのである．このLANのバックボーンは他のイーサネットかもしれないし，FDDIサブネットワークかもしれない．

2.2.2 イーサネットのパケット形式

図2-3はイーサネットバージョン2のパケットの基本形式を示している．発信元アドレスおよび宛先アドレスはいずれも48ビットの一意の識別子である．これらの識別子は全世界的に管理され，イーサネットコントローラカードの製造会社に対して管理が委託されている．これによりすべてのノードが一意の識別子をもつことが保証される．アドレスを使い果たしてしまうことを心配する読者がいるかもしれないが，46ビットの一意識別子は70 254 592 000 000のアドレスに相当する．

プリアンブル		
宛先アドレス		48ビット
発信元アドレス		
プロトコル型		有効な長さではない数
データ		データ+パッド≧46
パッド		
フレーム検査列		

2-3　イーサネットパケット

識別子の最初の2ビットはフラグとして用いられるため，個々のアドレスの一部としては用いられない．最初のビットはアドレスが個別のものであるかグループアドレスであるかを示す．グループアドレスは，単一のイーサネットサブネットワーク上のすべてのNISネームサーバというように，多数の局を参照するために用いられる．すべてのビットが1のアドレスはブロードキャストであり，1つのイーサネットの上のすべてのノードで受信される．ブロードキャストの使用はサブネットワークにまたがるアプリケーションでは一般に役に立たないことや，ブロードキャストを見境なく用いることはネットワークを輻輳させることから，次第に奨励されなくなってきている．

識別子の2番目のビットは，アドレスが局所的に管理されているか全世界的に管理されているかを示している．全世界的なアドレスは一意であることが保証されている．しかし，あまり賢いやり方とはいえないが，ネットワーク管理者がアドレスを割り当てることで，アドレス空間を局所的に管理することも可能である．この方式が最適ではないと考えられている理由は，（誰かが一意のアドレスを考え出さなければいけないということはもちろんのこと）アドレスが重複しやすいためである．

発信元および宛先アドレスの後には，プロトコル種別標識(protocol type indicator)が続く．この標識は，そのパケットがイーサネットサービスのどの利用者に対するものかを示している．プロトコル種別標識により，例えばDECnetとTCP/IPが単一のイーサネットのサービスを共有することができるのである．イーサネットモジュールはプロトコル種別のフィールドを見て，各々のパケットをしかるべき利用者へと配送する．プロトコル種別は長さとして有効な値であってはならないという図2-3の注に注意されたい．この注釈は，802.3形式について説明すれば明確になる．

イーサネットパケットの長さは60バイトと1524バイトの間でなければならない．イーサネットパケットの最小長は，衝突が起きたときにそれを検出するのに十分なだけ媒体上に存在することを保証するものである．

パケットのデータ部分はネットワークのイーサネットモジュールに対して透明である．いかなるデータを含んでいてもよい．代表的には，ネットワーク層のようなある利用者のデータを含ん

でいる．そのデータが，今度はネットワーク層のヘッダと何らかのデータ（トランスポート層のヘッダと何らかのデータ）から構築されている．

パッドはデータフィールドがイーサネットの最小長の要求を満たすようにするために空白を埋めるのに用いられる．パッドの後にはフレーム検査シーケンス（frame check sequence : FCS）が続く．FCS は送信されたデータに基づいて計算され，パケットの終わりに付加される．宛先ノードは FCS を再計算して，受信したものと比較する．イーサネットは通常誤りを起こさない媒体であり，FCS がくい違うことはあまりない．

2.3 IEEE の各種 LAN

イーサネットフレームフォーマットの一部はプロトコルに固有のものである．トークンリングなどの他の LAN 技術では，イーサネットパケットのすべての情報が必要とされるわけではない．例えば，トークンリングではパッドフィールドは必要ない．

IEEE が LAN 技術の標準化を開始したとき，イーサネットは以下の 2 つの副層に分けられた．

・論理リンク制御(Logical Link Control : LLC)

・媒体アクセス制御(Medium Access Control : MAC)

論理リンク制御は通信媒体に独立であり，トークンリング，トークンバス，CSMA/CD，FDDI などのすべての標準化された LAN 技術に適用される．LLC 副層の下で物理層の上に位置するのが MAC 副層である．MAC フォーマットはそれぞれの LAN 技術ごとに独自のものがある．

図 2-4 は IEEE802.3LAN パケットのフォーマットを示している．プリアンブル，宛先と発信元アドレスがイーサネットレベル 2 と同一であることに注意してほしい．これらはフィールドの MAC 部である．アドレスフィールドの後ろには長さフィールドが続く．イーサネットのプロトコ

プリアンブル		
宛先アドレス		48ビット
発信元アドレス		
長さ		
宛先SAP		802.1SNAPまたは実際のSAP
発信元SAP		
PDUタイプ		XID
プロトコルID		DSAP=SNAPのときのみ
データ		
パッド		
フレーム検査列		

2-4 802.3パケットフォーマット

ル種別フィールドが，長さとして有効でない数値でなければならなかったことを覚えているだろうか？ これによってパケットを受け取ったノードはそのパケットがどのような種別のイーサネットパケットかを決定することができ，これに続くフィールドを解釈することができるのである．

長さフィールドのあとには，宛先サービスアクセスポイント(SAP)標識が続く．宛先 SAP はパケットの LLC 部の始まりであり，すべての MAC 技術に共通である．このフィールドはイーサネットレベル 2 のプロトコル種別フィールドに似ている．

SNAP（subnetwork access point）として知られる特殊な SAP がある．SNAP は，データフィールドの最初のバイトがユーザのアドレスを表し，ユーザのデータは第 2 バイトからであることを示す．もし，SAP が本当のアドレスを保持しているのであれば，データの最初のバイトからユーザデータが格納される．発信元サービスアクセスポイントは，フレームを送出したノード上のプログラムのアドレスである．

最後に，ヘッダの LLC 部にはプロトコルデータ単位(PDU)種別フィールドが格納される．論理リンク制御には 2 つの方式がある．多くのネットワークで使用される種別であるクラス I の場合は，動作はコネクション状態をもたない．各々のパケットは独立した情報の単位である．

動作のもう 1 つのモードは，コネクション指向(connection-oriented)の LLC クラス II である．コネクション指向モードでは，送信されたのと同じ順序でデータのストリーム中のすべてのパケットが宛先 SAP に届けられることが，データリンク層の責任によって保証される．つまり LLC クラス II ではエラー検出と回復が行われる．

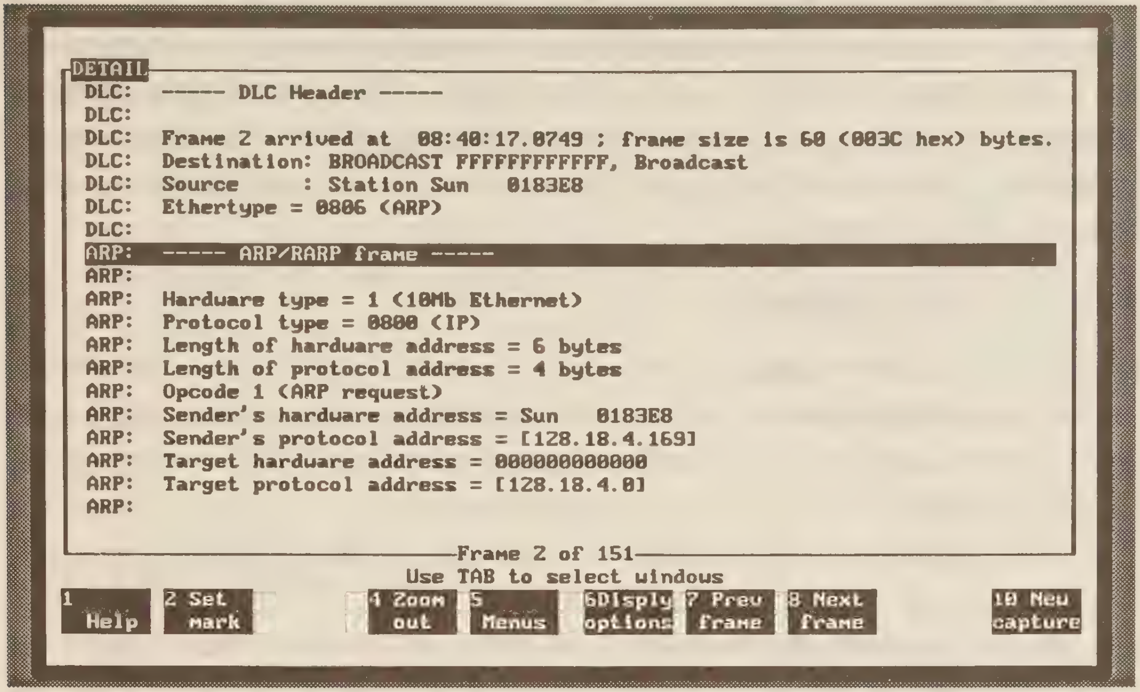
LLC クラス I は配送を最大限努力するサービスである．パケットは配送されるが，特定のパケットが確実に到着することや，2 つのパケットが送出されたのと同じ順序で到着することについての保証はない．

LLC クラス I 動作には，3 種類のパケットが使用される．上位層サービスユーザによって送られるフレームは，たいていは番号なし情報(Unnumbered Information : UI)である．交換 ID (Exchange ID : XID) とテストフレームは初期化と維持管理処理のために使われる．

2.3.1 イーサネットトラフィック

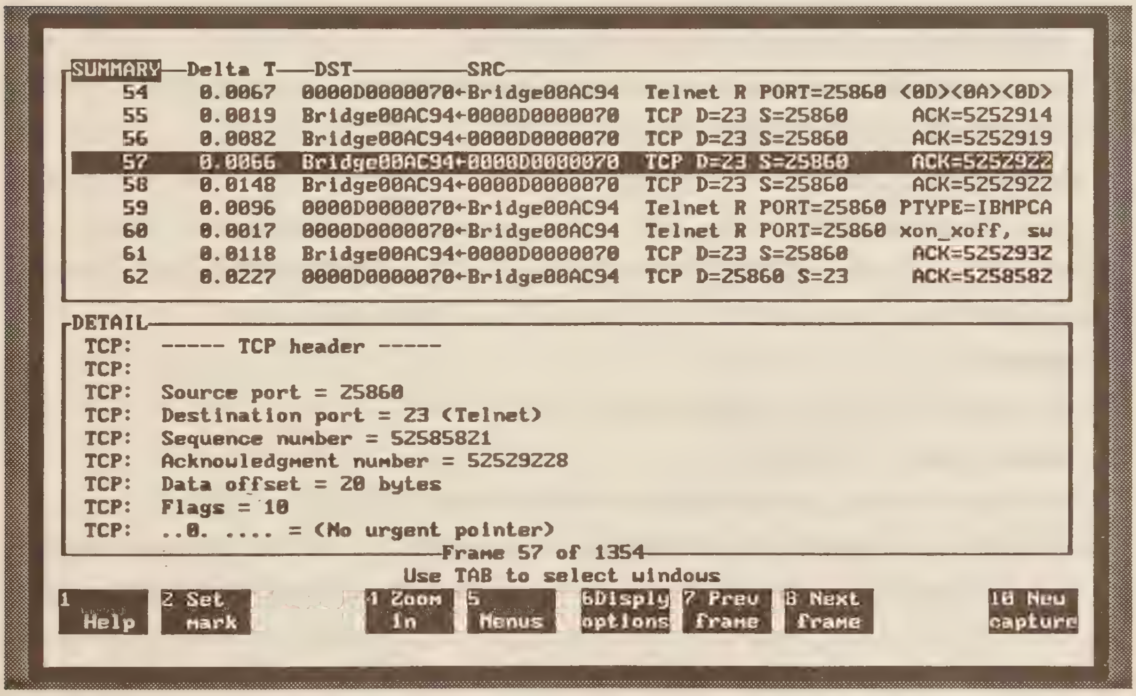
図 2-5 と 2-6 は，Network General 社の Sniffier Analyzer で記録された典型的なイーサネットトラフィックを示している．Sniffer は各フレームを解読して，異なる層のパケットのヘッダを示してくれる．

図 2-5 はパケットのデータリンク層がブロードキャストアドレスであることを示している．そのフレームの個々の発信元があり，その後にプロトコル ID がある．この場合，プロトコルはアドレス解決プロトコルである．パケットのデータリンク部の後にはデータフィールドが続く．データはイーサネットコントローラにとっては透過であり，コントローラはデータをそのままネットワークにブロードキャストする．宛先ノードが存在するかどうかや，宛先ノード上に宛先ユーザ



Courtesy of Network General

2-5　イーサネット上での ARP トラフィック



Courtesy of Network General

2-6　イーサネット上の複数のユーザ

(Ethertype)が存在するかどうかは，ネットワークのこの層で云々される問題ではない．

図2-6は，イーサネットが，異なるユーザのためのトラフィックを運ぶデータの高速道路として，いかにして機能するかを示している．この図には実際，イーサネット層の上にある複数の層が示されている．行き来しているトラフィックはTelnet仮想端末サービスと伝送制御プロトコル(TCP)である．どちらのサービスも下にあるインターネットプロトコルを利用しており，さらにこれがイーサネットを利用している．

2.3.2 イーサネットの基本構成

イーサネットの装置への接続は，イーサネットコントローラカードから始まる．イーサネットのもともとの通信媒体は同軸ケーブルであり，ThickWireないしベースバンドとして知られる．ケーブルへのアクセスはトランシーバと呼ばれる装置を用いて行われ，トランシーバはワークステーションなどの装置にトランシーバケーブルを用いて接続される．

もともとの通信媒体はベースバンド同軸ケーブルであるが，他の様々な通信媒体もIEEEによって認定されている．これらには次のようなものがある．

- 10BASE5(ThickWire)は標準的な同軸ケーブルである．単一のセグメントは最大500メートル，100ノードからなる．
- 10BASE2(ThinWire)はより細い同軸ケーブルであり，最大長は短いが，安価で取扱いも容易である．
- 10BASETはシールドされていない撚り対線を用いる．
- ファイバは機密を必要とするアプリケーションのために用いられる．また，ファイバはリピータやブリッジのためにも幅広く用いられている．
- 広帯域環境でもCSMA/CDプロトコルを利用することができるが，多くのサイトではむしろファイバの方へ移行している．

個々の通信媒体をどのように構成するかの詳細については，この本の守備範囲を越えているが，典型的な構成について述べると，以下のようないくつかの要素からなっている．

- 通信媒体
- ワークステーションやサーバ内のコントローラ
- 通信媒体と接続するためのトランシーバ
- トランシーバをコントローラに接続するために用いられるトランシーバケーブル

多くの実装では，通信媒体との1つの接続点に多くのコンピュータをつなぐためにマルチポートトランシーバを用いる．マルチポートトランシーバは集線装置(concentrator)と考えることができる．

ワークステーションのコントローラは，周辺機器バスの種類(つまりコンピュータの種類)や定常スループット速度によって様々なものがある．また，コントローラは，使用できる物理信号線

の種類によっても区別される．ThinWire を使用できるコントローラの多くは，ThinWire を撚り対線につなぐ BALUN（平衡/非平衡変換器）と呼ばれる装置を用いることによって，シールドされていない撚り対線（UTP）でも使用することができる．

標準的なコンピュータの他にも様々な種類の特殊装置が，イーサネット上でサーバとして機能することができる．イーサネットに存在する特殊サーバの 2 つの代表はプリンタサーバと端末サーバである．端末サーバは，多くの端末をネットワーク上のいくつかの異なるホストへ接続させるための安価な方法である．端末サーバは端末だけでなくモデムやプリンタを接続することもできる．

次の章で扱うルータは，いくつかの異なるイーサネット（または他のサブネット）を互いに接続させる 1 つの方法である．ルータは，扱うことのできるコネクションの数や，それがもつスループットの大きさによって異なる．他の装置と同様，ルータはいろいろなベンダから入手することができる．

2.3.3 マルチセグメントイーサネット

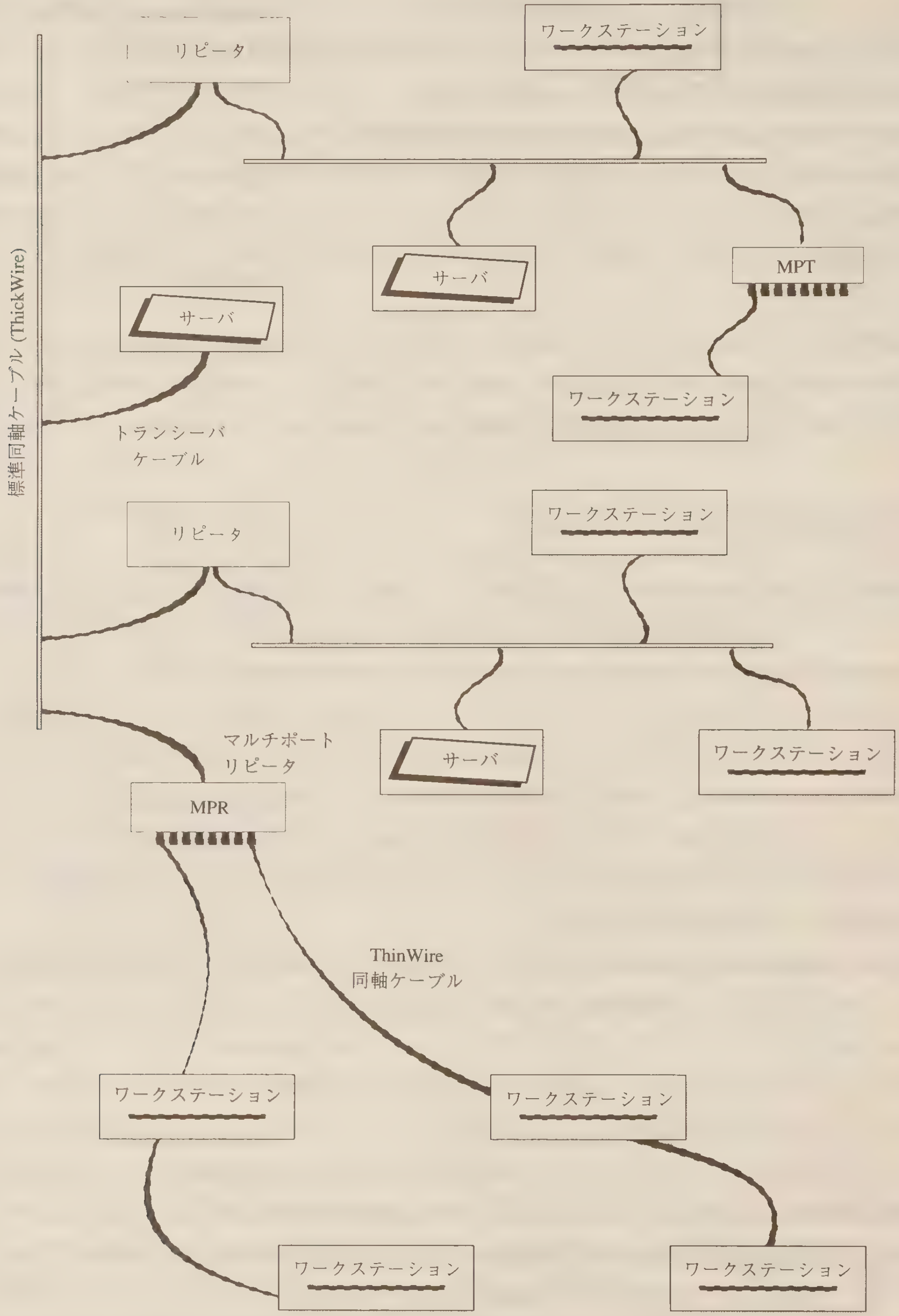
イーサネットの単一のセグメントは通常 29 ないし 100 ノードに限定される．29 は ThinWire イーサネットセグメントの典型的最大値（最大は 30 だが 1 つはリピータのために残される）であり，100 は従来から使われている同軸ケーブルのセグメントでの典型的最大値である．しかし，イーサネットアーキテクチャ自体は 1 つのサブネットワーク上で最大 1024 までのノードを扱えるようになっている．これを実現するためには複数のセグメントをリピータで互いに接続すればよい．

図 2-7 は，リピータで互いに接続されたマルチセグメントイーサネットを示している．リピータの役割は，リピータの片側からすべての信号を取り込み，タイミングをとり，増幅し直してからもう一方のセグメントへ再送出することである．リピータは，このようにして信号線をネットワークの物理層レベルで論理的に拡張する．

構成を行うにあたっての基本的な規則は，ネットワーク上の任意のノード間を結ぶ経路中に入れることのできるリピータの数が最大 4 台までであるということである．これからおのずと，典型的なマルチセグメントイーサネットはバックボーンとそこから枝分かれしたセグメントによって構成される，ということになる．異なるセグメント上のノードがデータのパケットをやりとりするには，2 つのリピータを通過することになる．

リピータ 2 台則は，イーサネットプロトコルでアーキテクチャ的に定義されているパラメータの範囲内で衝突が検出されることを保証するための手段以外の何ものでもない．トランシーバ，リピータ，通信媒体自体を含むすべての装置は，ネットワークに遅延を発生させるのである．

リピータ 2 台則のような規則は，過大な遅延が発生することを防ぐことによって，イーサネットが正しく機能することを保証するための手段である．このような規則は，通信媒体の種類や規則を定めるベンダごとに様々に異なる．例えば DEC の場合，イーサネットの配線方法は DECcon-



2-7　マルチセグメントイーサネット

nect によって規定されている．注意すべき点は，このような規則はあくまでも可能な方法の 1 つにすぎず，どの規則もイーサネットプロトコルが正しく動作する配線方法を用意することを目標としていることには変わりがないという点である．DECconnect などのいずれの配線システムも，実際にはベンダがたまたまそのときに売っていた一群のハードウェアのための販売用パラソル (marketing umbrella) にすぎないのである．

様々な種類のリピータが入手可能である．典型的なリピータは，2 つの ThickWire を 2 つのトランシーバケーブルを用いて互いに接続する．マルチポートリピータは，多くの場合 1 つの ThickWire (バックボーン) と最大 8 セグメントの ThinWire ないしシールドされていない撚り対線とを接続するのに用いられる．

第 3 の種類のリピータはファイバリピータであり，これを用いると 1000 メートル以上離れたセグメントどうしを接続することができる．ファイバリピータは実際には 2 つのハーフリピータからなり，それぞれが 1 つのセグメントに接続される．これらのハーフリピータどうしがファイバで接続される．

2.4 イーサネットのブリッジ接続

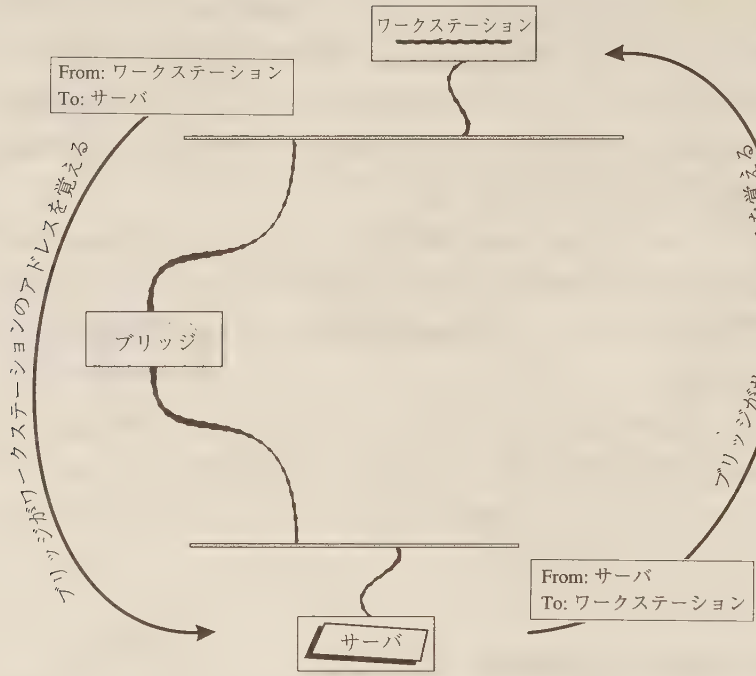
ブリッジは，複数のサブネットワーク（代表的にはマルチセグメントのイーサネット）を透過的に接続して，拡張サブネットワークを構成する装置である．拡張サブネットワークの利用者は，任意のノードに対して 1 パケット分のデータを送信することができるという基本的な性質を享受することができる．

ブリッジとリピータの違いは，ブリッジが中継 (forward) しなければならないパケットだけを中継するのに対して，リピータは物理層で動作するので，すべての信号変調を再送出する点である．したがって，ブリッジは発信元アドレスと宛先アドレスに応じてパケットを選択的に送信するフィルタといえる．中継を行うために，ブリッジはどのノードがどちらの側にあるのかを学習しなければならない（図 2-8 参照）．

ブリッジは初期化されると，各ポートを聴取し，各々の側にどのような発信元アドレスがあるのかを記憶する．この情報から，ブリッジはいくつかのノードの位置を知ることができる．しかし，すべてのノードが初期化の期間中に情報を送出するわけではないので，ブリッジがすべてのノードの情報を得ることはできない．

宛先アドレス，発信元アドレスとも同じ側にあるパケットに対しては，ブリッジはパケットを中継する必要がないので無視する．宛先アドレスが反対側にある場合には中継する．

未知の宛先アドレスに対しては，ブリッジは常にパケットの“洪水 (flood)”を起こす．すなわち，すべてのポートに対して，そのパケットを送出する（ここではブリッジがいくつものポートをもつと仮定している．もっとも，代表的なイーサネットブリッジは 2 つしかポートをもたない

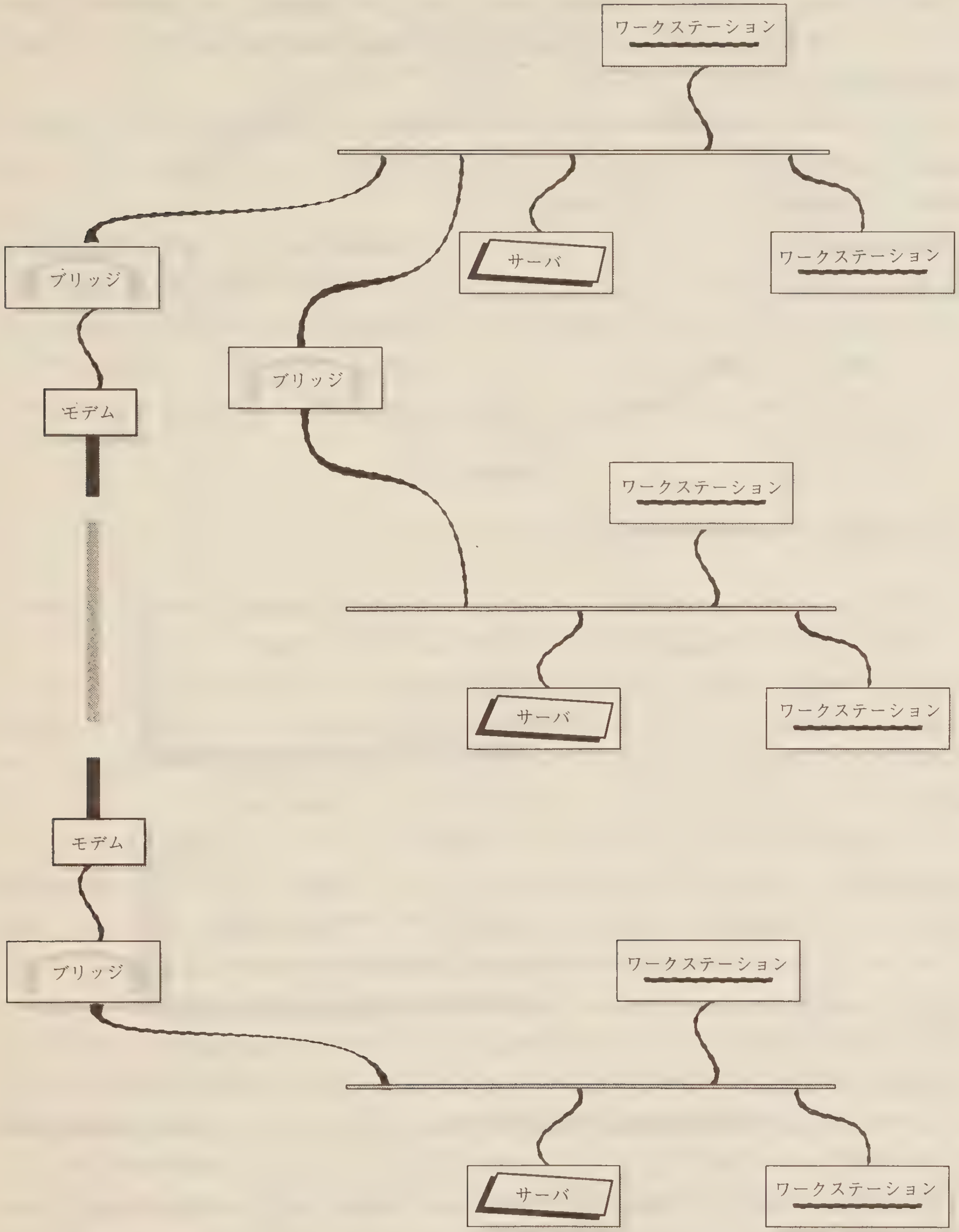


2-8　ブリッジの学習

が)．しばらくすれば，溢れさせたパケットの宛先が応答を返すであろう．その応答はそのノードのアドレスを発信元アドレスとしてもつので，ブリッジは目的ノードの位置を格納する中継表を更新することができる．このようにイーサネットパケットの発信元を調べることで，ブリッジはノードの位置を知ることができるのである．ここで，ノードは常にブリッジの片一方に存在すると仮定している．もし，ブリッジのトポロジーにループが存在すれば，この方式ではうまくいかない．

ループが存在しないことを保証するため，ブリッジは，イーサネットを接続しているのが一般的なメッシュではなく連結木(spanning tree)となることを保証するような，連結木アルゴリズムを用いる．もし2つのブリッジがループを構成するような場合には，そのうちの1つが無効化される．ブリッジによっては，“火の入った(hot)”待機状態となって，動作中のブリッジが故障したときに交代できるようにしている．もちろん，ネットワーク管理者はイーサネットを木構造に構築するはずなので，通常はトポロジー中のループが問題となることはない．

2つのノードの間にいくつものブリッジが存在することがあり得る(図2-9参照)．実際，インターネットワーク全体をブリッジを用いて構成することも可能である．ループが存在しない限り，ブリッジはパケットを中継し続け，利用者からは拡張イーサネット全体が1つの大きなサブネットワークのように見える．この方式の問題点は，ブリッジの用いるルーティング方式が単純であ



2-9　拡張されたイーサネット

るという点である．この本の以下の部分において，ネットワーク層がはるかに洗練された方式を用いて，ノード間の複数の経路(path)を追跡し，それらの経路の相対性能に応じてパケットの通り道を適性化することを述べる．

ブリッジの多くは，特定の種類のフレームだけを中継するフィルタ機能をもっている．例えば，Telnetの会話的なトラフィックはあるブリッジを通り抜けるが，ファイル転送はルータもしくは別のブリッジを通るというようなことが可能である．

プロトコル形式によるフィルタリングに加えて，多くの環境においては，通常，ブロードキャストを通過させない．ブロードキャストはすべてのノードからの返答を必要とするかも知れない．マルチセグメントのイーサネットでは関与するノードの数は数百であるのに対して，拡張イーサネットでは数千となり得る．もしブロードキャストの結果としてすべてのノードが返事をブロードキャストするようなことになれば，データの嵐(storm)はネットワークの能力をあっという間になくしてしまうだろう．

2.5 FDDIとトークンリング

イーサネットは10Mbpsで動作する．データリンク層と衝突回避機構のオーバヘッドを含めると，定常スループットが5Mbpsを越えることは稀である．ファイバ分散データインターフェイス(FDDI)は100Mbpsで動作し，理論的には80Mbpsのスループットを達成することができる．

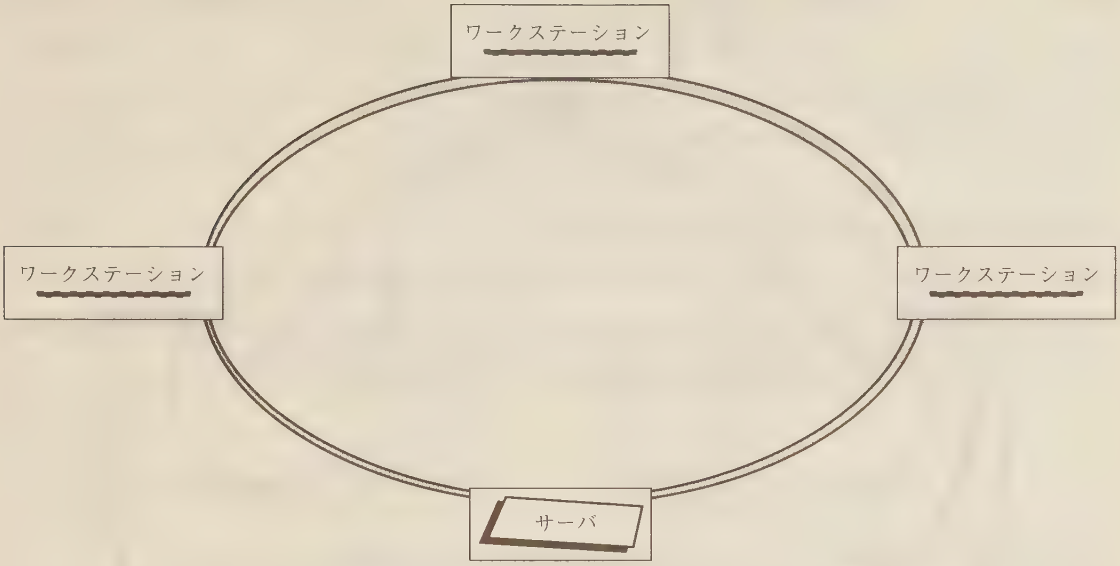
FDDIの基本構成はリングである(図2-10参照)．リング上の各ステーションは入力データを次のステーションに向けてコピーする．リング上のレシーバ(図2-11参照)は，データをコピーして流すだけでなく，自分のバッファへデータをコピーすることも可能である．

多くの場合，FDDIコントローラにはバイパス機能が含まれ，ステーションが動作していないときに使用される．一部のリンクが失われているときもバイパス機能によってリングの動作が維持される．バイパス機能は信号を再発生させず減衰させてしまうことに注意する必要がある．あまりに多くのステーションがバイパスモードになると信号は弱くなりすぎてしまう．

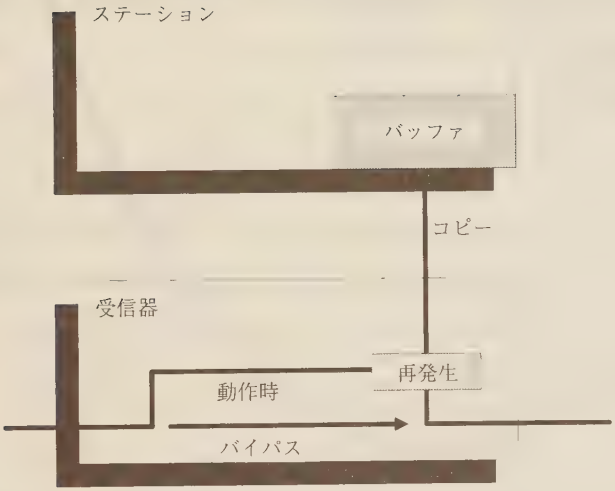
FDDIの基本動作は4ないし16Mbpsで動作するIEEE802.5トークンリングと同一である．しかし多くの場合，FDDIは2つのリングから構成される．一次リングは通常動作時に使用される．もし一次リングに障害が生じた場合は，データを迂回させるために二次リングが用いられる．二次リングが一次リングの障害の周囲に予備経路を設けることによって，ある程度の耐故障性が得られる．

FDDIの構成は，総ファイバ経路200kmに最大1000の物理リンクをもつことができる．二重リングが用いられた場合は，100kmの(二重)リングに500のノードがあるのと等価である．

FDDIの制限は，1.617msの最大リング遅延という1つのパラメータを元にして決められている．最大リング遅延は開始区切り文字がリングを1周するのに要する時間のことである．このパラ



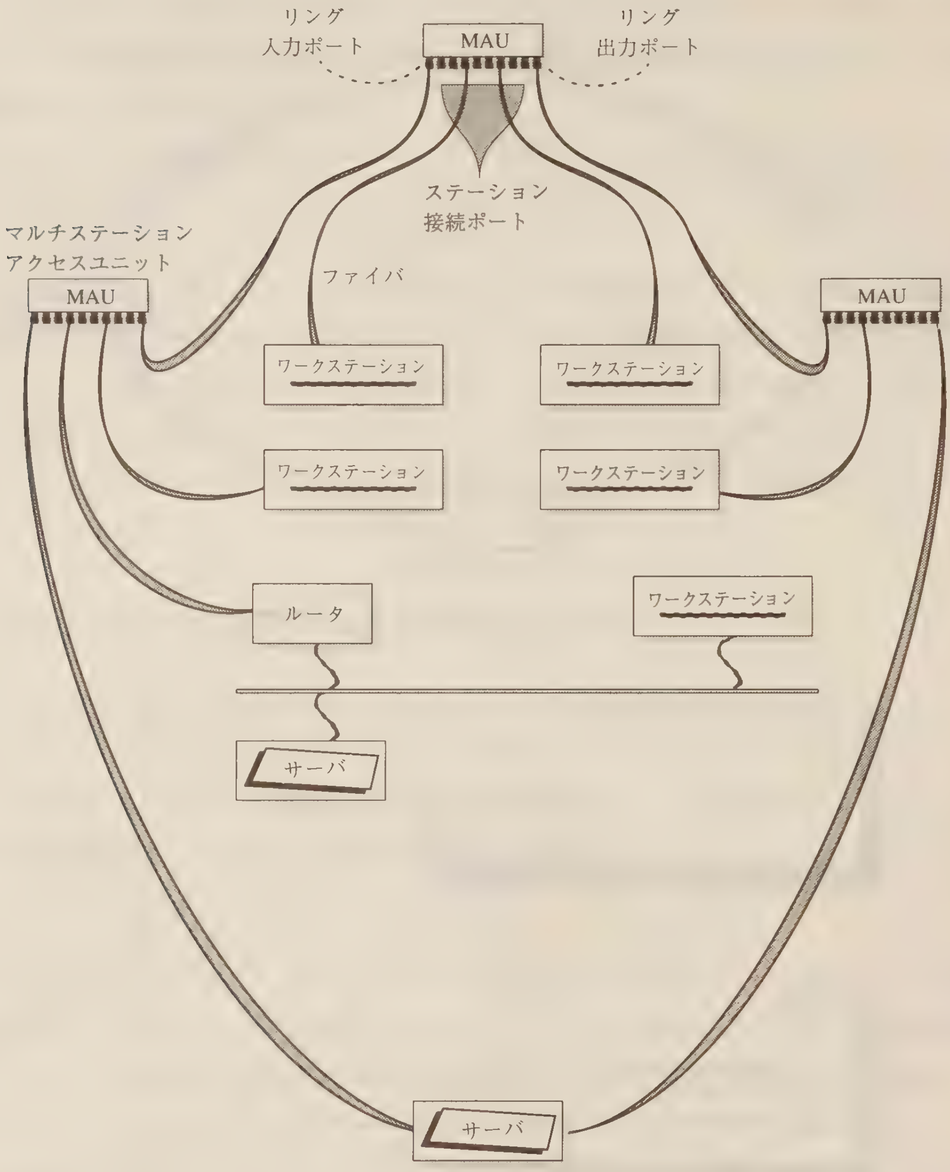
2-10 (とても)単純なリング



2-11 リングとの接続

メータを元にして，最大ノード数，最大距離，他の様々なリングパラメータが算出される．例えば総経路長が 200km であったとしよう．速度を 5085ns/km とすると遅延時間は 1.017ms となる．もし 1000 の物理接続がありそれぞれの遅延が 600ns だとすると，全体で 0.6ms の遅延が発生する．したがって，200km で 1000 接続のリングの総遅延は 1.617ms となり，最大リング遅延パラメータと一致する．

FDDI リングにノードを直接取り付けることも可能であるが，多くのベンダは集線装置としてマルチステーションアクセスユニット(MAU)を使用している．二重リングには MAU が直接つ



2-12　FDDIリング

ながり，MAU から(通常)8 ないし 16 本のポートが引き出される．ステーションが動作していない場合でも，信号減衰の問題は発生しない．なぜなら，バイパス機能は MAU 側で用意されるからである．

集線装置の 1 つの大きな利点は，ワークステーションやサーバに単一リングコントローラだけを用意する方が，二重リングを装備するよりも安価になることである．FDDI の仕様は，二重リング接続ステーション(クラス A)と単一リング接続ステーション(クラス B)の両方を定義している．図 2-12 は集線装置を基本としたリングの構成例を示している．いくつかのステーションは中

心のリングに直接接続されているが，多くのステーションはMAUに接続されていることに注意してほしい．

いくつかのステーションはリングからさらに遠ざけられてイーサネットに接続される．イーサネットはブリッジを用いてMAUに接続される．ブリッジの代わりにルータを使用することもできるであろう．

典型的なブリッジは，パケットからイーサネットヘッダを取り除いてから，FDDIヘッダを付加して再送出する．このような翻訳型(translating)ブリッジに対して，イーサネットヘッダの先頭にFDDIヘッダを追加するだけのカプセル型(encapsulating)ブリッジもある．カプセル型ブリッジはデータの宛先もイーサネットである場合にのみ機能するが，翻訳型ブリッジは様々なMAC技術上で動作できる．

2.5.1 FDDIの動作

イーサネットの場合は，通信媒体が空いていればどのノードもデータを送出することができるが，FDDIのようなトークンリングの場合，ノードがデータを送出できるのは，トークン(図2-13参照)と呼ばれる特殊なパケットを受信したときだけである．トークンはリング上をノードからノードへと渡されていく．もしノードに送出したいデータがなければトークンは隣にコピーされるだけである．もし送出したいデータがある場合はトークンは取り込まれる．取込み手順は，トークンのコピーを行わずにアイドルシンボルを書き換えることである．

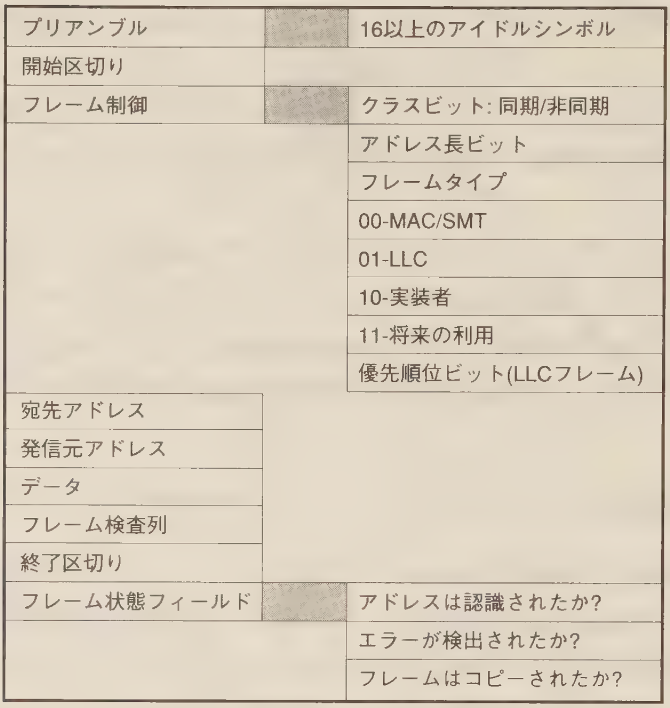
次にノードはトークン保持タイマ(THT)によって管理される一定時間を与えられ，リング上のトークンを置き換えることを許される．この期間にノードは1つ以上のデータフレームを送り出すことができる．制約トークン(restricted token)を用いると，2つないしそれ以上のノードが別のノードに割り込まれる心配をせずにトークンのやりとりをして会話することができる．

データフレームは送出されるとリングを周回する．おそらくは，いずれかのノードがアドレスを見て，データを隣のステーションにコピーするだけでなく自分のバッファにもコピーするはずである．フレームが送出したノードまで戻ってきたら，データをアイドルシンボルに置き換えることによってそのフレームをリングから取り除く．

図2-14はデータフレームのフォーマットを示している．フレームの最後にはフレーム状態フィールドがあり，送信先ノードがアドレスを認識したかどうかを知ることができるようになっている．もしアドレスが認識されたらこのビットは反転される．もしフレームをコピーするのに十分

プリアンブル		16以上のアイドルシンボル
開始区切り		
フレーム制御		制約トークン?

2-13　FDDIトークン



2-14 FDDIフレーム

なバッファ領域があれば，フレームコピー済みビットも反転される．もしフレーム検査手順に失敗した場合はエラー検出ビットが用いられる．

データフィールドは理論上は約4500バイトに制限されるが，多くのFDDIの利用者はずっと小さなパケットサイズを利用するだろう．このデータフィールド内にはLLCヘッダがあり，その後に上位層のヘッダが続く．

フレーム制御フィールドはリングの基本動作を制御するために用いられる．例えば，アドレス長ビットは2バイトアドレスか標準の6バイトアドレスかの選択を行う．フレームタイプは，そのフレームがFDDIステーション管理(SMT)機能で用いられる内部MAC管理フレームであるかLLCデータフレームであるかを示す．フレームタイプ，優先度ビット，クラスビットに関するより詳しい情報は以下の節で示す．

2.5.2 動作のモード

FDDIリングは以下のような2つのモードで動作する．

・同期モードは各ノードに対し，ある程度のバンド幅と応答時間を保証する．

・非同期モードはバンド幅の動的な共有を可能にする．

非同期モードはトークンによって即座に割り当てられるが，同期バンド幅はあらかじめSMTプロトコルを用いて割り当てておく．初期のFDDIの実装では同期モードは使用されていない点

に注意する必要がある.

同期バンド幅は，ネットワーク全体の総バンド幅の部分に対応するトークン周回目標時間(TTRT)のうちの何パーセントを使うか，という形で割り当てられる．いうまでもないことだが，同期割当ての合計は 100%以下である.

ゼロでないバンド幅が割り当てられた各ノードは，トークン周回タイマを開始させることについて心配せずに，一定時間データフレームを送出することが許される．もし，すべてのノードが同期送出を終了した後で，まだバンド幅の余裕があれば，ノードはトークン保持タイマの限度まで非同期送出を行うことができる.

非同期送出では，バンド幅の割当てはさらに 2 種類に分けられる:

・無制約モードはデータの送出を希望するすべてのノードに時間スライスを与える.

・制約モードは 1 つの拡張会話のために占有させる.

通常は無制約モードで動作する．このモードでは，トークンの割当ては優先順位法を用いて行われ，これはトークンがリングを周回する時間に基づいて決められる．各優先順位レベルごとに，トークン周回時間(Token Rotation Time : TRT)のしきい値が決められている.

TRT が大きくなると，低い優先順位レベルは切り捨てられる．ノードは，より高い優先順位のデータだけしか送出できなくなる．高優先順位でトークンがリングを周回していると，ついには，すべてのノードがその最高優先データの送出を終える．その結果トークンはリングをより高速で周回するようになり，低優先順位のデータフレームにも送出される機会が与えられるようになる.

ターゲットトークン周回タイマの値はリングで得られる総バンド幅を表す．同期送出が終了した後で残されているバンド幅は，トークン周回タイマという形で与えられる．現在の TRT と目標 TRT(TTRT)との差が，利用できる非同期バンド幅を表す．TRT が最小の値であれば，それは誰も同期トラフィックを出さなかったことを意味する.

無制約トークンが受信されると，2 つのノードは制約モードに入る．第一のノードは最初のデータの塊を送ってから，制約トークンを発行する．受信ノードは自分のデータを送ってから制約トークンをリングに送り返す．制約モードは(隣どうしの ID の交換などの基本ステーション管理タスクを含む)すべての無制約非同期トラフィックを禁止する．制約モードの会話に入る，終了させる，続行させるの判断は FDDI を利用する上位層に任される．同期送出はトークンに影響されないので，制約モードにも影響されない.

ステーション管理の機能の 1 つに，制約モードの最大時間をとり決めることがある．制約モードの場合には，トークン保持タイマに従う必要がないということに注意する必要がある．その本来の性質により，制約モードは他のノードに対する公平性をすでに先取りしているからである.

制約モードと同期モードの使用は FDDI のオプション機能である．おそらく制約モードはシステム管理などのきわどい機能のために用いられるであろう．同期モードは音声などの実時間アプリケーションや，ノードの遠隔立上げなどの時間的にきわどいアプリケーションのために用いら

れるであろう.

2.5.3 リングの初期化と回復

リングの初期化の必要性を検出すると,ステーションはクレームトークンフレームを発行する.クレームフレームを送出し続けることによって，複数のステーションがリングの初期化を先導する権利を取り合う.

次に，各ステーションは受信クレームフレームを見て，そのフレームが自分の送ったものか他のステーションのものかを判断する．それぞれのフレームには入札値が書かれており，この値が最大のものが勝つ.

最大入札値が衝突した場合は，トークンがリングを連続的に周回するのに要する時間を示すために，トークン周回目標時間(TTRT)が用いられ，これに基づいて調停が行われる．TTRT が最小のものが入札に勝つ．もし TTRT 最小のものが複数あった場合は，ステーションアドレスが最大のものが勝つ.

あるステーションがクレームトークンプロセスに勝つと，直ちにリング初期化プロセスを開始する．まず，トークン周回目標タイマに，成功したクレームフレームの値を設定する．次に，トークン周回タイマ(問題の生じたトークンを検出するのに用いる)をリセットし，無制約トークンを発行する.

このトークンがリングを 3 周してから,通常のデータトラフィックが可能になる.各ステーションは最初の 1 周で TTRT の値を設定し，リングが再び動作状態になったことを示す内部フラグを立てる. 2 周目で同期トラフィックが発行される. 3 周目で非同期トラフィックを開始させることができる.

クレームトークンプロセスは各ステーションで時間が計測される．一定時間が経過しても入札が解決していなければ，ステーションはビーコンを送信する．まだ入札過程にあるステーションだけがビーコンを送信できることに注意する必要がある．入札に敗れたステーションにおいて時間経過が検出されたときは，そのステーションは入札プロセスに再参入する.

リング中のどこで問題が発生したかの正確な位置決定を行うために，ビーコンフレームが用意されている．ビーコンは，ステーションが動作しなくなったかもしくは通信媒体に問題が生じたかのいずれかの理由により，リング中に重大な論理的障害が発生したことを意味する．ビーコンを発信したステーションは継続的に発信を続ける．クレームトークンの場合と同様に，受信されたすべてのビーコンは検査される．もし受信されたビーコンのステーションアドレスが自分のものと異なれば，ビーコンを発信する上流のステーションがあることを意味するので，そのノードは発信をやめる.

各ノードがなびいていくに従って，ビーコンの発信源はリングの障害に近づいていく．あるとき，リングが修復されると，ノードは自分自身のアドレスが書かれたビーコンを受信するように

なる．これはリングが再び動作状態になったことを意味するので，クレームトークンプロセスに入る．

2.6 トークンリングのサポート

トークンリングネットワークはSunワークステーションを接続するためのサブネットワーク手段としては用いられないが，トークンリングをネットワークの中に統合することは可能である．IBMのネットワークは，パーソナルコンピュータの場合も，より大きなシステムの場合も，トークンリングサブネットワークに基づいていることが多いため，このことは重要である．

トークンリングシステムの環境では，3種のネットワークのうちの1つが用いられている．まず，IBMのSNAが用いられていることがある．この場合には，TCP/IPの世界とSNAの世界はゲートウェイを介して接続される．第2の可能性は，トークンリングにおいて，NovellやBanyan，LAN Managerのような何らかのパーソナルコンピュータネットワークが用いられている場合である．パーソナルコンピュータネットワークの多くは，TCP/IPをサポートしている．Novellのように，NFSに至るまでのプロトコルを明確にサポートしているものさえある．第3の可能性は，単純にネットワーク全体でTCP/IPを用いることである．PC-NFSや，様々なIBMのミニコンピュータやメインフレームのソフトウェアパッケージを用いることで，この本で述べるすべてのプロトコルを用いることが可能である．

トークンリングでどのようなプロトコルが用いられているかにかかわらず，トークンリングをTCP/IPに接続するための簡単な方法は，ゲートウェイとなるシステムを用いることである．ゲートウェイシステムは，どんなゲートウェイもそうであるように，ネットワークごとに1つの，少なくとも2つのアダプタをもっている．Sunの製品でこれを行うものはSunNet TRI/Sアダプタであり，Sバスでトークンリングを利用することができるようになる．これ以外にも，イーサネットを直接に，あるいは距離を隔ててトークンリングに接続するためのいろいろな製品がある．

他のシステムもゲートウェイとして用いることが可能である．例えばSun-3またはSun-4をゲートウェイとして用いる場合には，Formation fv1600という，4Mbps，16Mbpsの両方のトークンリングを利用できるVMEインターフェイスがある．IBM PCシステムをゲートウェイとして用いることも可能である．

2.7 広域リンク

イーサネットとFDDIはSunネットワークのための一次的なLAN技術である．他にもHYPERchannel，Ultranet，HIPPIリンクなどいろいろあり，この章の最後のケーススタディで紹介されるであろう．しかし，ここではいくつかの広域用技術について調べていこう．

広域環境で用いられるプロトコルには2つのクラスがある．まず，X.25などのプロトコルに基づく公式な広域データネットワークがある．これ以外の多重アクセスデータネットワークにはフレームリレー，SMDS，ISDNやこれらと同様なプロトコルがある．もう1つのクラスのプロトコルは専用のポイント間リンクのために用いられるものである．たった2台のコンピュータが通信するだけであり，仮想回線やアドレスづけなどの概念が不必要であるため，これらのプロトコルははるかに簡単である．

TCP/IPネットワークのために用いられる基本プロトコルが2つある．最も簡単なものはSLIP，シリアル回線IPである．これのより現代的な後継者はポイント間プロトコル(PPP)である．

2.7.1 SLIP

シリアル回線インターネットプロトコルは非標準であることで知られる．皆が利用しているが，事実上の標準にとどまっている．これは，プロトコルが単純なため，機能させるために実際にはほとんど何も必要としないことが理由である．SLIPの典型的な利用は1200bpsないし19.2kbpsでのモデム接続である．

SLIPは3ComのUNET TCP/IP実装で導入された．SLIPはアドレスづけもパケットタイプの識別もエラーの検出や補正も行わず，単にデータだけを送るパケットフレーミングプロトコルである．SLIPはRick AdamsによってUnix4.2BSDに付加され，Sunワークステーションにも付属されたため，ホストと遠隔ルータとを接続する方法としてたちまち広く使われるようになった．4.3BSDの標準部分であると同時に，Ultrix，SunOS，端末サーバ，IBMのPC用のものもある．

SLIPでは2つの特殊文字が使われる．エンド(END)とエスケープ(ESC)である．これらの制御文字が正規のデータストリームから透明となるように，簡単な文字詰込み方法が使用されている．もしEND文字を普通のデータとして送りたいときは，ESCと8進の334に置き換える．もしESC文字を普通のデータとして送りたいときは，その代わりにESCと8進の335を送る．

パケットを送るためには，データとEND文字を送ればよい．上位互換性のある簡単な拡張として，始めと終わりにEND文字を置くことがPhil Karnによって提案されたことは注目に値すべきことである．これによって雑音で誤りが生じた行のバイトを読み飛ばすことができる．

読み飛ばすべき雑音行がなければ，ノードは隣合せの2つのENDを受信する．これはIPからは長さゼロのパケットに見えるので，捨てられるだけである．SLIPの実装のレベルで捨ててしまってもよい．

SLIPの実装は，事実上の標準に過ぎないために，互いにうまく動作することもあればうまくいかないこともある．しかし，多くの場合，うまく動かない唯一の原因は，最大パケットサイズに関するオプションが異なることである．バークレーの実装では1006バイトを扱うことになっており，事実上の標準における最大転送単位(MTU)の標準値の役割を果たしている．

2.7.2 PPP

SLIPにはいくつかの実際的な限界があり，そのうちの1つで無視できないものに回線1本当り1人のユーザしか使えないという問題点がある．異なる上位層のユーザたちのためにアドレスが用意されたりしてはいない．SLIPのより現代的な代替品としてポイント間プロトコル(PPP)がある．このプロトコルに関する仕事は1988年に始まり，1990年までに相互運用のデモンストレーションが始められた．

PPPは，RS-232-C，RS-422，CCITT V.35などのよく使用される標準を含む，どのようなDTE/DCE物理インターフェイス(通信ネットワークを利用する端末と装置の間の物理リンクを表す用語)上でも動作する．唯一の要求は全二重回線を必要とすることである．制御信号は特に要求されてはいないが，実装を行うときにはそれらがあった方がよいであろう．

データリンク層で，PPPは同期もしくは非同期回線として動作する．フレーム化のやり方はISO高水準データリンク制御(HDLC)プロトコルと大変に似ている．HDLCでのフレーム化手法は国際標準なので，ハードウェアで処理されることがよくある．PPPは標準的なHDLCフィールドのどの値を使用するかについて記述しており，さらに追加のプロトコルフィールドをもっている．3つのプロトコルのグループがある．リンク制御，ネットワーク層プロトコル制御，そしてネットワーク層プロトコルデータである．

PPPはIP層とその下にある物理層との間に位置する．リンク動作のときにPPPの果たす役割はとても簡単である．各々のPPPパケットはIPパケットを保持する．しかし，標準的な隠蔽機能を提供することによって，TCP/IPソフトウェアは下に位置する物理リンクの詳細から分離される．

PPPリンクの確立は4つの段階からなる．まずリンクが確立され，適切な構成についての取決め(negotiation)が行われる．取決めの対象となるオプションとしては以下のものがある．

- 最大受信サイズ
- 非同期制御文字の割当て
- 認証方式
- 暗号化方式
- リンク品質監視用パラメータ

構成処理の後で，認証フェーズを実行してもよい．これは，単純なユーザ/パスワードプロトコルでもよいし，多くの場合そうであるように何もしなくてもよい．PPPは電話を用いたIPネットワークの主要プロトコルであるため，認証は重要である．

次に，リンク品質決定を行ってもよい．PPPは，リンクが使いものになるかならないかをノードがどのようにして決定するかについての方針を規定はしていない．その代わりにPPPには，様々な方針(例えば誤り率やスループット)に利用できるように，エコー要求と応答の機能が用意

されている.

リンクの監視は，いくつのパケットとオクテットが失われたかを決定するという方法に基づいて行われる．両端で送出ないし受信されたパケットとオクテットの数が数えられ，次にその情報が別の側で利用できるように送る．両端でこのデータの状態を同期させるための工夫がなされている.

同期はリンク品質監視(LQM)パケットをチェックポイントとして利用することによって行われる．こちら側の値とともにLQMが送られる．向こう側で計数値が受信されると，バージョンとともに調べられる．両者を比べることにより，失われた数が判明する.

最後に，ネットワーク層プロトコルが立ち上げられて，リンクはリンク準備完了状態となる．IP制御プロトコル(IPCP)はIPを動作可能にしたり停止させたりするために用いられるプロトコルである．ここでの取決めのために用意されたオプションには，IPアドレスや圧縮方式などがある.

IPアドレスオプションは，ノードにIPアドレスを検査したり割り当てたりすることを可能にする．このフィールドは2つの副フィールドからなっている．当方IPアドレスと先方IPアドレスである．パソコンが電話でIPネットワークにつながったような場合，ノードがつながった特定のポートに応じてアドレスが変化するため，自身のIPアドレスをあらかじめ知っていることができないかも知れない．このオプションによって，遠隔ノードがそのセッションにおけるパソコンのアドレスを決定することができる.

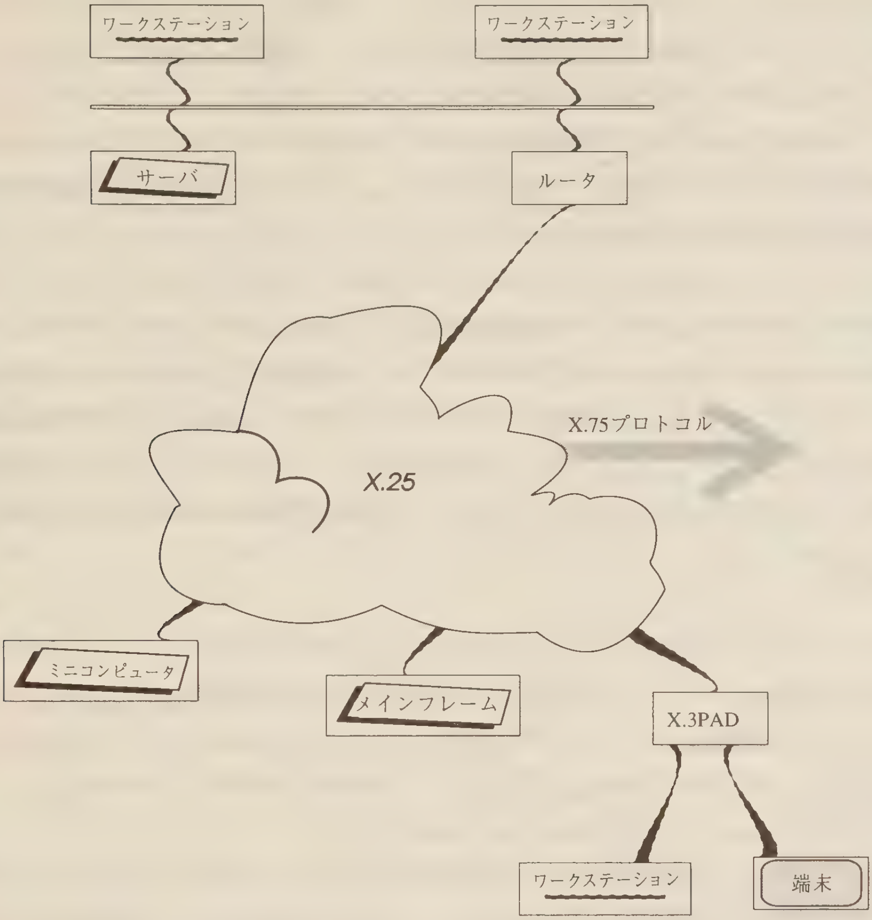
圧縮は，Van Jacobsonによって用いられているヘッダ圧縮方法を利用することを可能にする．電話回線を介してつながっているX-ウィンドウやTelnetセッションを考えてみよう．どちらのパケットもその大部分はIPやTCPのヘッダからなる．Van Jacobsonのヘッダ圧縮法は，各パケットの利用率を高め，低速回線での効率的な高スループットをもたらしている.

2.7.3 X.25

データリンク層において，X.25プロトコルはHDLCのサブセットとなっている．さらに，X.25は物理層とネットワーク層での機能も定義している．X.25のネットワーク層の機能は，X.25サブネットワーク内の2つのノード間で仮想接続を確立するために用いられる．次に，LAPBプロトコルによって，どのようにしてX.25ユーザがネットワークと整合されるかが定義される．実際にX.25は様々なネットワークの異なる部分で使用される.

X.25はコネクション指向である．ネットワーク層のユーザは遠隔目的地への仮想回線を確立させる．仮想回線は適当な場所に永久的に確立されている場合もあるし，動的に確立させることもできる(交換仮想回線).

図2-15は典型的なX.25の構成を示している．X.25はネットワークとのインターフェイスを示しているだけであることに注意する必要がある．内部処理は実装に依存するので，雲として表



2-15　X.25の構成

されている．

X.25は様々な目的で利用することができる．ホスト対ホストの通信では，双方のコンピュータ上に高水準プログラムが存在する．このようにして，イーサネット上のルータがミニコンピュータやメインフレームや別のルータへの仮想回線を確立することができる．次の章では，X.25が複数のLANを結合する手段として利用できることが示される．

X.25は端末がホストと通信する手段でもある．X.25プロトコルは同期データ転送を定義しているが，多くの場合端末は非同期である．パケット組立/分解装置(PAD)として知られる装置が，非同期装置を同期X.25ネットワークにインターフェイスするのに用いられる．PADの動作はX.3 CCITT標準の中で定義されている．

端末(または端末をエミュレートするワークステーション)とホストがどのようにして互いに通信するかを定義するために，さらに2つのプロトコルが用いられる．X.28プロトコルは端末が，どのようにしてPADとインターフェイスされるかを定義する．X.28は，PADがいつ受信データ

を取り込み，X.25パケットに組み入れるかなどについて定義している．X.29プロトコルは，ホストがどのようにPADと通信して端末を制御するかについて定義している．

X.25のアクセスは様々な方法によって提供されている．SunLink X.25や他のゲートウェイシステムがそうであるように，多くの場合はルータにイーサネットボードとX.25インターフェイスの両方が装備される．これらのゲートウェイの多くは，X.28やX.29による端末の使用とネットワーク層のIPからの利用の両方を許している．

X.25インターフェイスは様々な状況で重要である．銀行の支店などのように多くの目的地をもった私的なネットワークは，いくつかの賃貸回線を使いながらその上にX.25ネットワークを構築したいかも知れない．逆に，公衆X.25ネットワークは臨時のアクセスのために使われる．ひとたび，1つのX.25ネットワークに接続すれば，別のX.25ネットワーク上の目的地に到達することもできる．X.75プロトコルは，1つのX.25ネットワークがどのようにして別のネットワークに相互接続するかを定義している．このようにして，交換仮想回線は国境を越えて確立される．

2.7.4 SMDS

マルチメガビットデータ交換サービス(SMDS)は主に主要都市間ネットワーク(MAN)を対象とした，非常に高速なネットワークである．またSMDSは長距離通信業者からWANサービスとして提供されている．SMDSは様々な速度で動作することができるが，最初の試行接続はDS-1(1.544Mbps)またはDS-3(44.736Mbps)で行われる．SMDSは最終的には同期光ネットワーク(SONET)の最大150Mbpsないしそれ以上の転送速度を使用するようになるであろう．

SMDS標準は実質的にはIEEE802.6MAN標準と同じである．どちらも音声とビデオのための準備がなされているが，当初のプロトタイプはデータ通信のみ動作する．基本的な考え方は，T1からT3の速度でのアクセスを必要に応じてユーザに対して許すことであり，SMDSは交換型の(switched)サービスということになる．

SMDSは，多くのアプリケーションがバースト的であり，データ要求が時間的に大きく変動するという事実に基づいて設計されている．しかし，これはいつでも必ず正しいわけではない．例えば，音声はバンド幅要求がかなり一定である．

多くの異なる応用からなる環境(例えばTCP/IP)では，要求を測定することは困難である．最初の間はTelnetセッションのトラフィックだけであったのに，次の瞬間に数メガバイトのファイル転送が行われたりするからである．

SMDSは広帯域ISDN(BISDN)の先駆者として位置づけられている．BISDNは45Mbpsの倍数のバンド幅を提供し，基本チャネルは150Mbpsないし600Mbpsで動作する予定である．SMDSは，51.84Mbpsから13Gbps以上にわたる速度の光ファイバ伝送標準であるSONETと互換性がある(しかし依存しているわけではない)．相互乗入れネットワークトラフィックの60%以上はSONETに変更されるであろうとする予測がある．

2.8 SWAN：一例

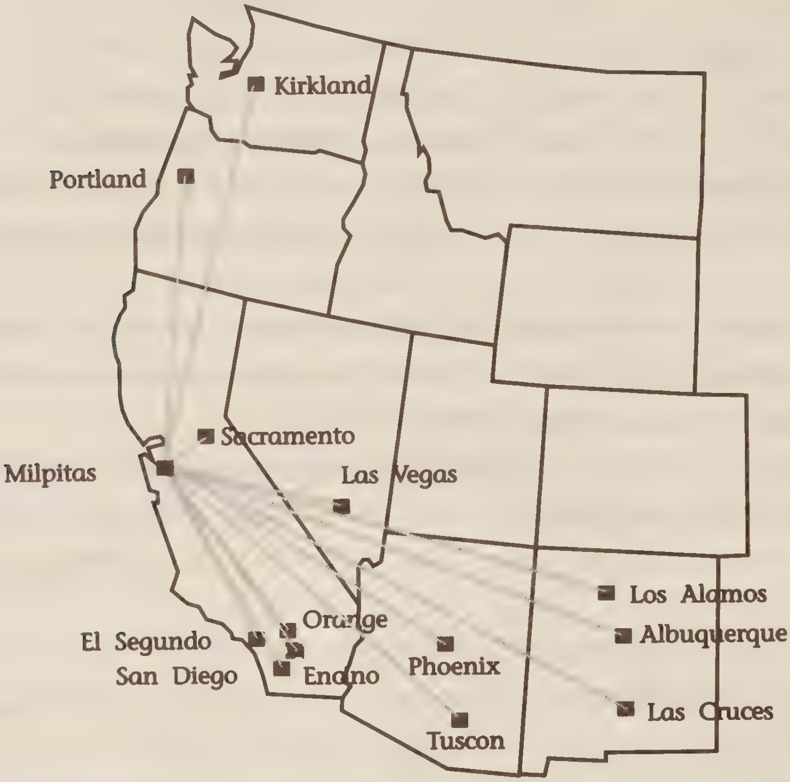
いろいろなインターネットワークがあり，その多くは国際 TCP（および OSI）インターネットワークである Internet に接続されている．いくつものネットワークをどのようにして接続するかを理解するため，まず Sun の企業ネットワークである Sun Wide-Area Network（SWAN）を見てみよう．

SWAN はすべて TCP/IP プロトコルに基づく，20 ヶ国の 10 000 ノード以上から構成されている．主たるプロトコル集は TCP/IP であるが，DECnet，Open Systems Interconnection（OSI），そして Systems Network Architecture（SNA）も役割を担っている．

米国内では，二重 T1 ネットワーク（T1 回線 1 本は 1.544Mbps の専用ディジタル回線）がネットワークの中核となっている（図 2-16 参照）．この T1 のバンド幅は T1 マルチプレクサを用いて音声，ビデオおよびデータに割り当てられる．イギリス，フランス，日本との接続には 56kbps の回線が用いられている．



2-16　SWANバックボーン



2-17 SWAN西部州地域

サンフランシスコベイエリア（Bay Area）では，Sun ネットワークの中核である DS3(45Mbps) ファイバ回線が Mountain View，Milpitas，Palo Alto の主要地区を結んでいる．2つの異なる Internet ゲートウェイにより外の世界へアクセスすることができる．

ハブとなっている都市はそれぞれの地域における分配点となっている．例えば図 2-17 はカリフォルニアの San Jose の近くにある Milpitas をハブとする西部地域を示している．Milpitas と Billerica は Sun にとってどちらもきわめて重要なハブである．Bellerica はヨーロッパへのゲートウェイであり，イギリスにあるヨーロッパハブとの間が専用回線と衛星によって接続されている．

Milpitas は極東へのゲートウェイの役割を果たしているとともに，Sun の多くの部署があるベイエリアのハブにもなっている（図 2-18 参照）．Milpitas は Menlo Park，Mountain View，Palo Alto にある他の主要な敷地と T3 の速さのファイバで接続されている．この地域にあるそれ以外の施設とは，T1 またはそれより遅い回線で結ばれている．

SWAN のバックボーン全体は，音声，ビデオ，そして（もちろん）データにわたる様々なアプリケーションで利用することが可能である．異なる種類のトラフィックを 1 つに束ねるため，T1 マルチプレクサが用いられている．図 2-19 にコンピュータ機器を SWAN に接続するための典型的な方式を示した．



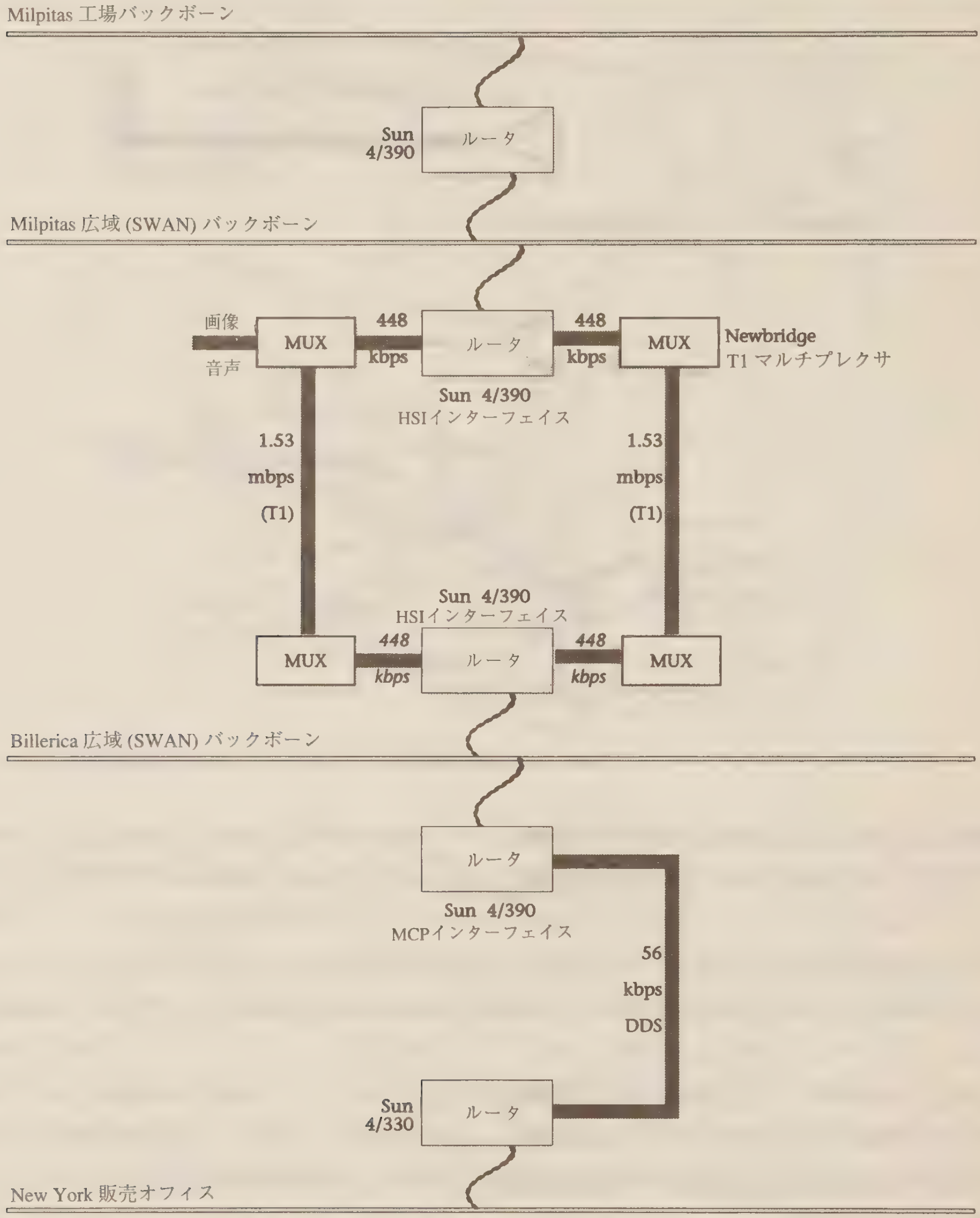
2-18 SWAN湾岸地域バックボーン

Milpitas と Billerica の地域の間のすべてのデータを経路づけするために，2 台の Sun-4/390 が用いられている．それぞれのハブには SWAN のバックボーンがあり，現在のところそれらはイーサネットである．これらのイーサネット上にユーザはなく，ルータのみが存在する．

例えば Milpitas では，SWAN バックボーンと Milpitas 工場のバックボーンをつなぐルータがある．そのバックボーンが今度はいろいろな建物や研究室にある小規模なイーサネットを接続している．

このようにトラフィックを分割しているのは，性能とセキュリティの両方の理由による．例えばヨーロッパから入ってきてテキサス州のヒューストンに行くトラフィックは SWAN バックボーンだけを通り抜ける．Milpitas の建物に行き着くトラフィックだけが Milpitas バックボーンへとルータを通り抜ける．

Milpitas と Bellerica の間の主 SWAN 回線は，サーバ以外に 2 つの設備を用いている．T1 マルチプレクサは異なる種類のトラフィックを束ねるのに用いられている．回線のデータ部分を駆動するのに用いられているのは Sun の高速シリアルインターフェイス（High Speed Serial Interface : HSI）通信コントローラというボードであり，この場合，これらの地点の間の T1 回線のうちの 488kbps の速度の回線を提供している．



2-19　SWANバックボーンにリンクするデータ

Billerica と地域内の販売店との間では，これほどの性能は必要としないので，より安価なマルチプロトコル通信プロセッサ（Multiprotocol Communications Processor : MCP）を用いて，ディジタル専用回線上の 56kbps の回線を駆動している．ニューヨークにあるルータがオフィス内の

バックボーンに接続されていて，そこにワークステーションやサーバを接続することができる．

このネットワークにより，Sunのどのユーザも他のどのユーザとも通信することができる．このネットワークは非常に大量のトラフィックを処理している．システム上には10 000人以上の社内ユーザがいて，20ヶ国への接続がある．3つの異なるInternetゲートウェイがInternetへのアクセスを提供しており，それに加えて多数のUUCPに基づく回線がある．

外部からのアクセスとして最も注目すべきものは，外部メール中継ホストであり，これは1日当り60 000メッセージを軽く処理する．このメールハブは個々のユーザにメールを配布するサブドメインのメールハブの間でのメールの移動を行っている．

大部分のメール配布は，TCP上の単純メール転送プロトコル(Simple Mail Transport Protocol：SMTP)を用いて内部的に行われる．これはユーザを見い出すのにドメイン名前システム(Domain Name System：DNS）を用いている．これに加えて，X.400に基づくメッセージ処理領域と接続するためのX.400回線もある．ファックスメール機能により，ユーザはウィンドウ上のfax toolという，mail toolによく似たユーティリティを用いてファックスによるメッセージを送ることができる．

ネットワークでは各種のファイルを利用できるように，NFSとオートマウンタが広く用いられている．多くのユーザは直結のディスクドライブがあってもウィンドウシステムを遠隔マウントしている．

メインフレームは2つの異なるSun Channel Gatewayを通じてアクセス可能である．IBMメインフレームへのアクセスがすべてSunワークステーションからなされるというのは面白い．すなわち，3270端末は用いられていない．メインフレームでの仕事の多くは，注文の記入や出荷状況のためにCullinet DBMSにアクセスすることである．この他のメインフレームのアプリケーションとしては，MRP(製造材料計画技術)および支払口座システムがある．システム上には平均500人から600人のユーザがいる．

NFSは広く用いられており，それにはMVSのデータをSunワークステーションにマウントすることも含まれる．ネットワーク中では1000台以上の大規模サーバが用いられている．米国東部やヨーロッパの技術部隊は，米国西部にあるソースシステムにアクセスするのに大陸横断のマウントを用いている．広域NFSアクセスの鍵は十分なバンド幅をもつことであり，この場合は二重のT1回線により提供されている．

いくつかのサーバは何千人ものユーザが利用できる．例えば，企業全体からアクセス可能な単一のバク（bug）データベースサーバマシンがある．ニュースサーバはもう1つの例であり，1台のマシンが技術部隊全体に対してニュースアクセスを提供している．

名前はいくつもの異なるNISドメインに組織化されている．複数のNISドメインを扱っているNISサーバもいくつもある．すべての名前，パスワード，そして別名は，単一の企業全体にわたるサーバに格納されている．マップは毎晩ドメインサーバすべてに配布される．マップはきわめ

て大きいことに注意してほしい．いくつかのものは1メガバイトよりも大きい．第9章で述べるypxfrユーティリティが，NISマネージャからグループへ情報を転送するのに用いられている．

ソースコードを有するマシンはすべてNIS名前サービスにより管理される大域オートマウンタマップに記載されている．企業全体に及ぶオートマウンタマップを用いることにより，ユーザは他のどのユーザのホームディレクトリに移動することもできる．もちろん，そのユーザがしかるべきNFS権限をもっているとしてのことだが．

NFSに加えて，RPCに基づくいろいろなプログラムが利用可能である．例えば，自動車のライト消し忘れ用のデータベースがある．企業内のすべての車のナンバーがデータベースに入っている．もしヘッドライトが点き放しになっている車を見つけたら，“light”というコマンド名とプレートナンバーを打ち込めば，車の持ち主にメールメッセージが送られる．

Sunで最もよく使われているRPCプログラムは相場サーバかもしれない．これは，Sun（および主要な競争相手）の株式の現在の状況を与えるものである．多くの重役たちは株をもっているので，しばしば彼らが最初に使い方を覚えようとするプログラムはこれである．主任技術者たちにおいても，しばしばStock Toolをデバッガや8進ダンプやviエディタの隣においている．

このネットワークの管理は領域ごとに分割されている．部署のコンピュータを管理する責任は個人(またはグループ)が負っている．中央にある設備やSWANについては中央にある情報資源(Information Resources : IR)グループが面倒を見ている．ネットワークを管理するのにはしばしばSunNet Managerが用いられている．特に，個々の装置を監視するのにSNMP, DECnet, NICE, IBM Channelエージェントが用いられている．

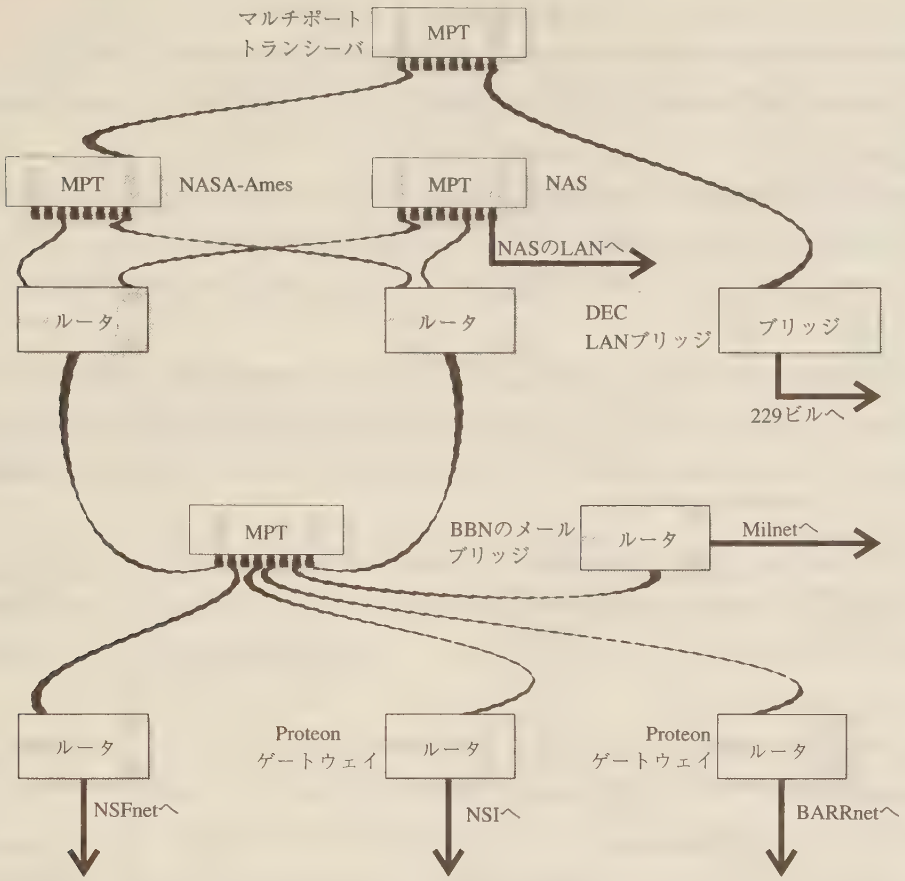
2.9 Internetについて

Sunエンジニアリングのユーザが，例えば回路設計問題をモデル化するために，ネットワーク上の研究施設のCrayコンピュータのサービスにアクセスしたくなったとしよう．この節では，2つの地点についての例を述べ，Sunが，Internetを形成する企業ネットワークや研究ネットワークのまさに1つであることを示すことにする．

SWANの各地域(region)をリンクするために用いられるSun Milpitas SWANバックボーンを覚えているだろうか?　このバックボーンには，ベイエリア地域研究ネットワーク（Bay Area Regional Research Network : BARRnet)に接続されているルータが含まれている．

Mountain Viewの技術者は，ファイルを別のネットワークに転送し，そこでスーパーコンピュータにファイルをコンパイルさせて実行させたいと思っているかもしれない．この処理のためにファイル転送プロトコルなどの適当なプロトコルが選ばれる．

初期処理が済むと，技術者のワークステーションからある外部IPアドレスをもったパケットが発行される．パケットは，いくつかの中間ルータを経由して，Sun Milpitasバックボーンに到達



2-20 NASA-Ames の広域リンク

するだろう．

そこで，BARRnet に接続されたルータは Internet アドレスが外部向けであることを知り，BARRnet 上にパケットを送出するであろう．このパケットの考えられる 1 つの目的地は，Mountain Viewの近くのNASA-Ames研究センターにある数値空気力学シミュレーション（Numerical Aerodynamic Simuration : NAS）センターである．

このようにしてパケットは，Sun の Mountain View の施設から Milpitas の SWAN バックボーンを通って，NASA-Ames の広域バックボーンに到達する．図 2-20 はこのバックボーンの構成の一部を示している．

NASA-Ames がハブとなっている主要ネットワークのための 3 つの主ルータ（もしくはルータの集合体）が用意されている．NSFnet は核となる全国的研究ネットワークであり，これに BARRnet などの地域ネットワークが接続されている．BARRnet から NSFnet へ向かうトラフィックの大部分はこのバックボーンを通過する．

バックボーンというのはケーブルの塊ではなく，マルチポートトランシーバ(“イーサネットの缶詰”)であることに注意する必要がある．ネットワークのこのレベルは，軍用ネットワークのMilnetやNASA自体のWANであるNASA科学インターネット(NASA Science Internet)へのリンクも含んでいる．

もし，NASの施設を使用する許可を得ていれば，パケットはこれらのルータのどれかを通って外に出ることはなく，NASA-Amesネットワークを通り続ける．NASA-Amesへのアクセスを提供する大きなゲートウェイが2つあり，そこからトラフィックはNASや他の研究施設へルートされる．

ルータの冗長性は有用である．なぜなら，ネットワークへの経路が常に用意されることを意味するからである．今度はこれらのルータがいくつかのNASA-Amesのバックボーンに接続される．そのうちの1つはNASのものある．図2-21はこのネットワークへ到達した後の構成を示している．

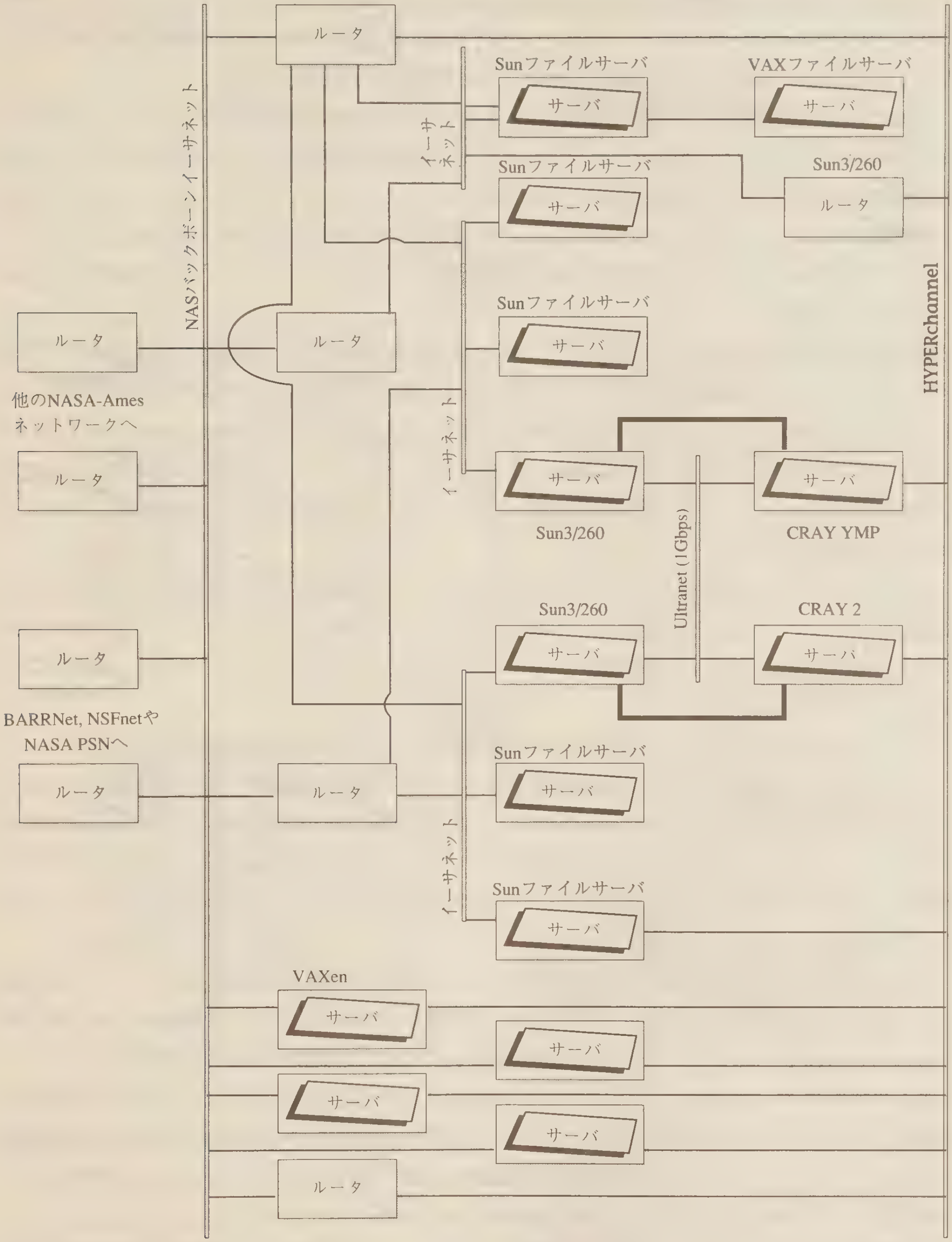
ネットワークからのトラフィックは図の左のNASバックボーンイーサネットに入る．ユーザは少なくともあと1つのルータを抜けて，どれかのサーバシステムに到達する．多くの場合，遠隔ユーザは，直接Crayコンピュータにログインして，Crayを高価な端末サーバとして使用するなどということはしない．

Internetから入ってきたユーザはVAXシステムの1つをフロントエンドシステムとして利用する．Mountain ViewのワークステーションからNASのVAXへのTelnetセッションを確立させる．新たなコマンドが準備されて，ここからCrayコンピュータや他のシステムに送られることになる．これで，われわれのユーザも他の所内のユーザと同じになる．とはいえ正確にはいろいろ違う点もある．われわれのユーザには端末セッションしかないが，所内のユーザはX-ウィンドウを走らせてスクリーン上にきれいなディスプレイを表示したりできる．いや，これもそうとは限らない．もし2つのノードの間のパスがある程度高速で，低位層のプロトコルでヘッダを圧縮するための技法が使われていれば，X-ウィンドウでさえもが広域環境で動作できる．

NASネットワークは一連のイーサネットシステムから構成されており，これらにローカルサーバが接続されている．これに加えて，ローカルユーザによって使用される多くのSunやSilicon Graphicsのワークステーションがある．

多くのサーバシステムはイーサネットとHYPERchannelの両方に接続されている．この中速度ネットワークは，ファイルシステムのマウントなどのサーバ間の通信のために用いられる．

2つの特別なサーバは2台のCrayコンピュータシステムである．これらのシステムはイーサネットに接続する代わりに，1Gbpsで動作するUltranetネットワークをもっている．この高速通信チャネルは，ネットワークファイルシステムを用いて何ギガバイトものディスク空間を2台のマシンの間でクロスマウントするために使用されている．大規模サーバへの高速な直接リンクをもった何台かの専用Sunワークステーションもある．



2-21　NASネットワーク

今度は，われわれの技術者がNASA-Amesへのアクセスをもっていなかったとしよう．その代わりに，国立のスーパーコンピュータセンター(その能力の何パーセントかを商用に用いることを許していることが多い)の1つに商用アカウントが用意されているとしよう．

それでも，われわれのパケットはBARRnetを経由してNASA-Amesのバックボーンに送られるかも知れない．しかし，NASネットワークに送り込まれる代わりに，われわれのパケットはInternetに送り出されるかも知れない．アカウントがサンディエゴスーパーコンピュータセンター(SDSC)のものならば(これはカリフォルニアの利用者のいかにもありそうな選択である)，パケットはNSFnetバックボーンを(T3の速度で)通過してSDSCバックボーンに到着するかも知れない(図2-22参照)．

SDSCはNSFnetのハブの1つであるが，NASA-Amesと同様に，他の多くのネットワークのゲートウェイとしても機能している．ルータを互いに接続するために，何階層ものマルチポートトランシーバが使われている．これらのルータの多くは，地域ノードへのDECnetないしTCP/IPトラフィックのために，T1ないし56kbpsリンクを提供している．これらのうちのあるものは顧客からのスーパーコンピュータアクセス専用のリンクであるが，別のリンクは地域のネットワークのために使われている．

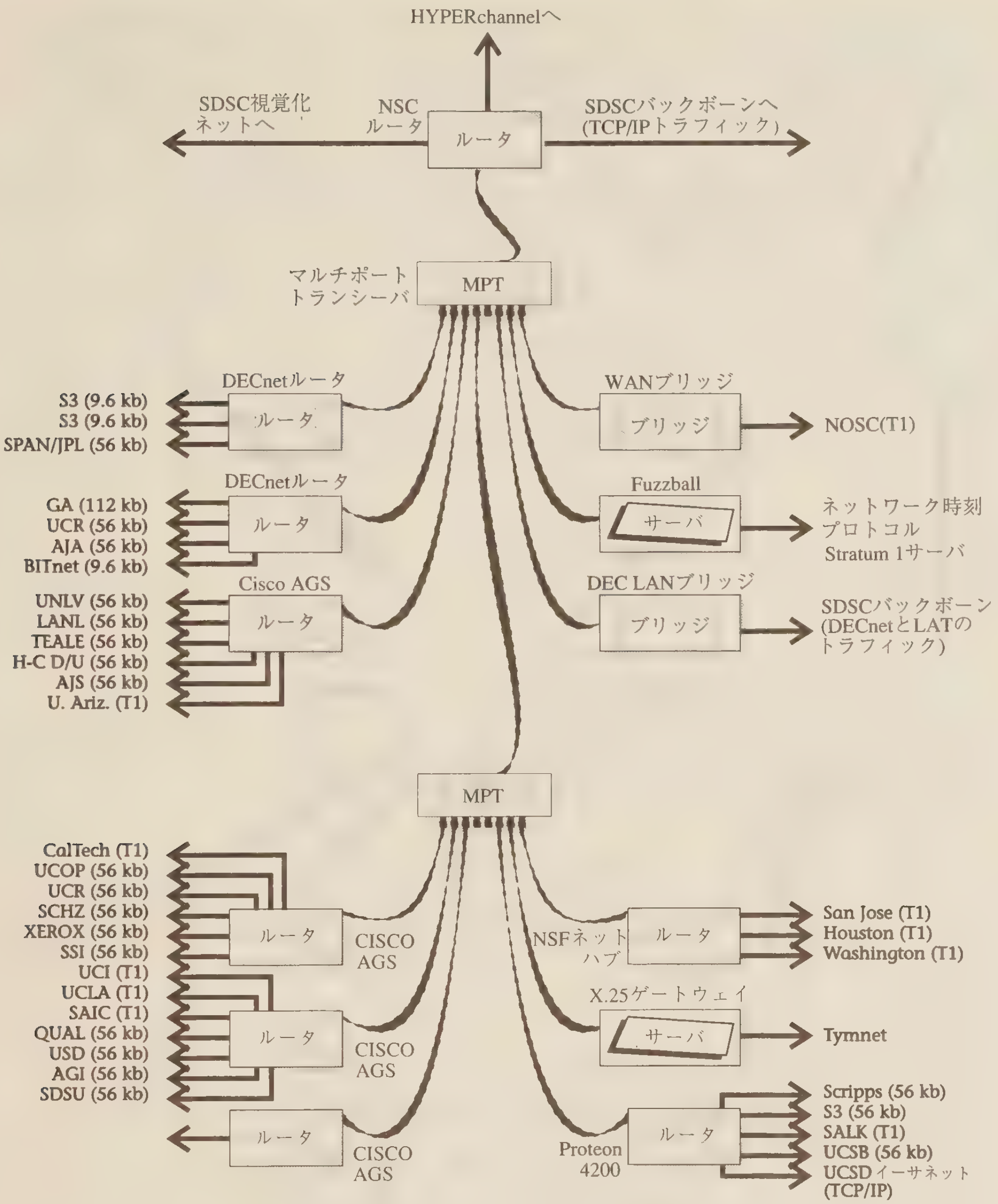
SDSCはNSFnetと他のネットワークの両方のハブとして機能しており，他のネットワークには地域ネットワークであるCERFnetが含まれている．これを図2-23に示す．各ハブがターミナルサーバをもっており，ユーザが簡単なターミナルでInternetのサービスにアクセスできるようになっている点に注意せよ．

しかし，今回見ているユーザはCERFnetの設備の1つを目指しているわけではないので，パケットは，図2-24に簡略化した形で示したSDSCネットワーク内を進み続ける．WANバックボーンは遠隔ユーザにとってのエントリーポイントである．結局，これらの遠隔ユーザの多くはWANバックボーンを進んで，図の下にあるSDSCイーサネットバックボーンにおりてくる．前の例と同様，ここでユーザは，フロントエンドサーバ(この例の場合は一連のVAXミニコンピュータ)を用いてTelnetセッションを開始したりする．

VAXシステムどうしは70MbpsのVAX Clusterシステムを用いて接続されている．これは遠隔ユーザに仕事場を提供し，その一方で，Crayが生成したアニメーションを映画のために取り込むのに用いられるフィルムレコーダなどの特殊出力デバイスを駆動する能力も提供する．

多くの所内のユーザは別のイーサネットを利用する．例えば，Crayコンピュータからの結果を視覚化しているユーザは，視覚化用ネットワーク上にワークステーションをもっているであろう．このネットワークは，一連のSunやSilicon Graphicsのワークステーション，いくつかのフレームバッファ，AlliantやVAXのサーバをもっている．

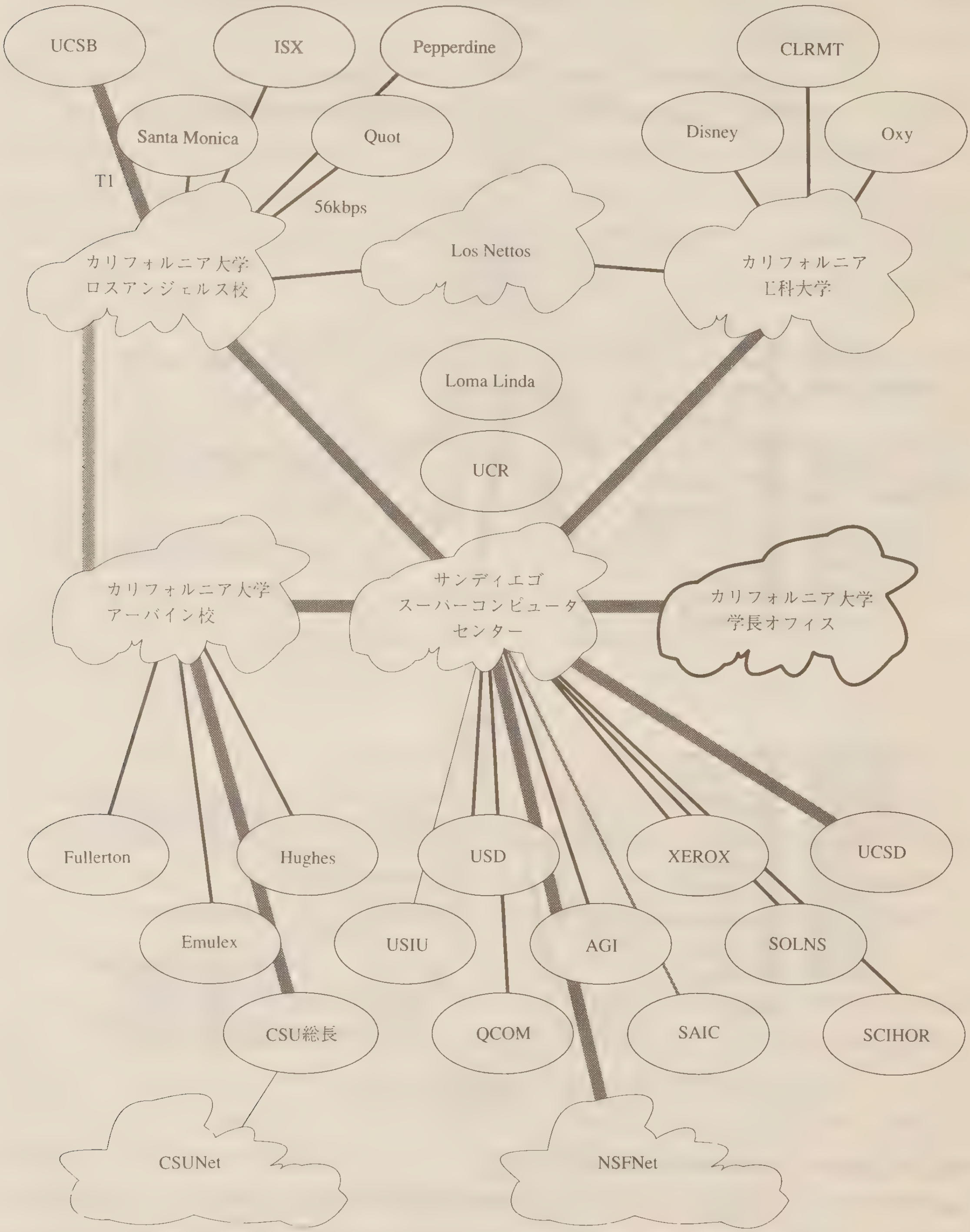
より強力な設備へのアクセスが必要な場合，トラフィックは50ないし100Mbpsの多重リンクで動作しているHYPERchannelシステムへ送られる．HYPERchannelは64ギガバイトの記憶空



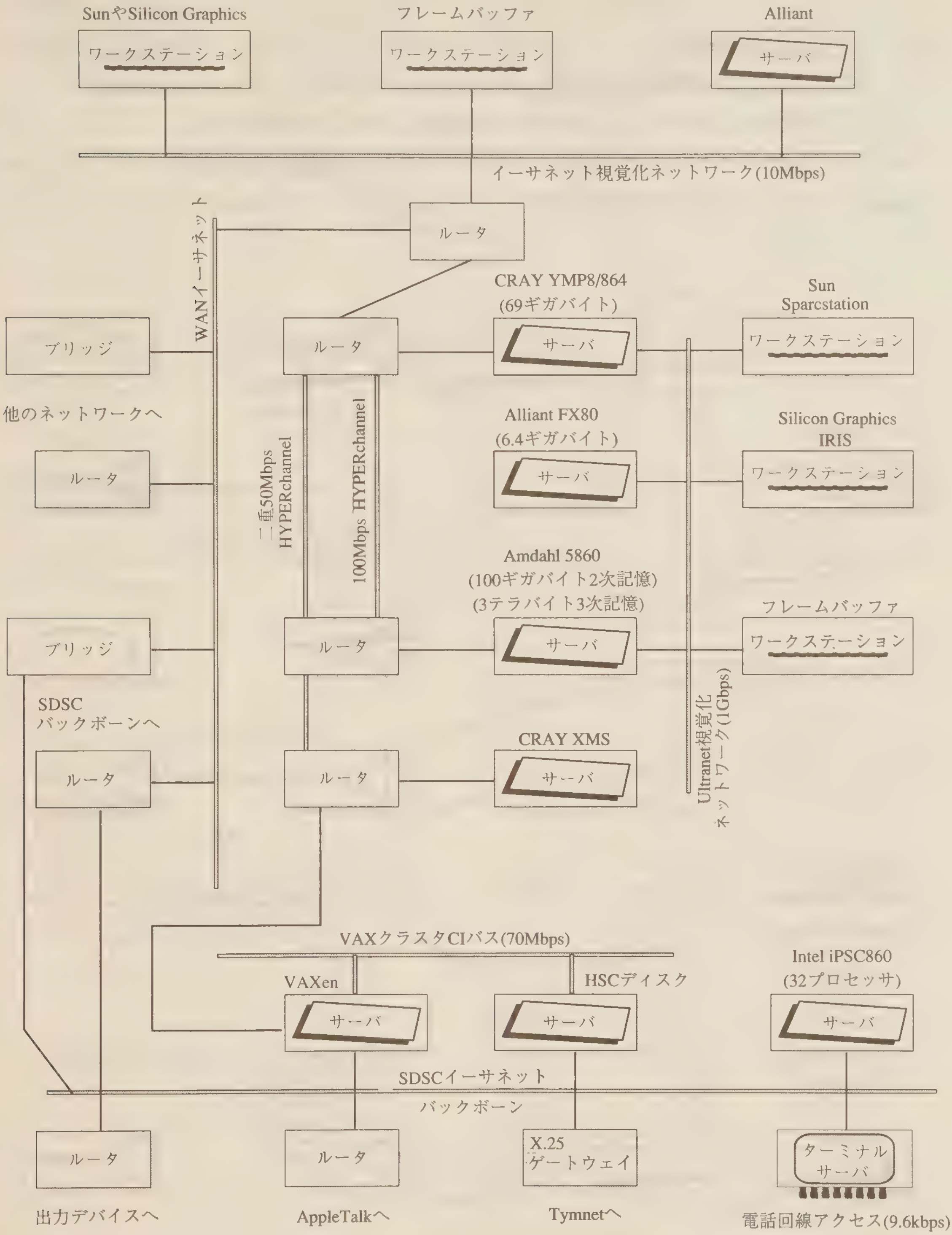
2-22 SDSC WANリンク

間をもったCRAY YMP8/864へ接続されている．

64ギガバイトはワークステーションにとっては膨大な量に見えるかも知れないが，Crayコンピュータにとってみればたいしたものではない．例えば，NASの設備は2ギガバイトの主記憶と2ギガバイトのRAMディスクをもっている．このような大きなシステムの食欲を満たすためにAmdahlメインフレームが記憶装置コントローラ専用に用いられている．


UCSB
ISX
Pepperdine
CLRMT
Santa Monica
Quot
Disney
Oxy
T1
56kbps
カリフォルニア大学
ロスアンジェルス校
Los Nettos
カリフォルニア
工科大学
Loma Linda
UCR
カリフォルニア大学
アーバイン校
サンディエゴ
スーパーコンピュータ
センター
カリフォルニア大学
学長オフィス
Fullerton
Hughes
USD
XEROX
UCSD
Emulex
USIU
AGI
SOLNS
CSU総長
QCOM
SAIC
SCIHOR
CSUNet
NSFNet

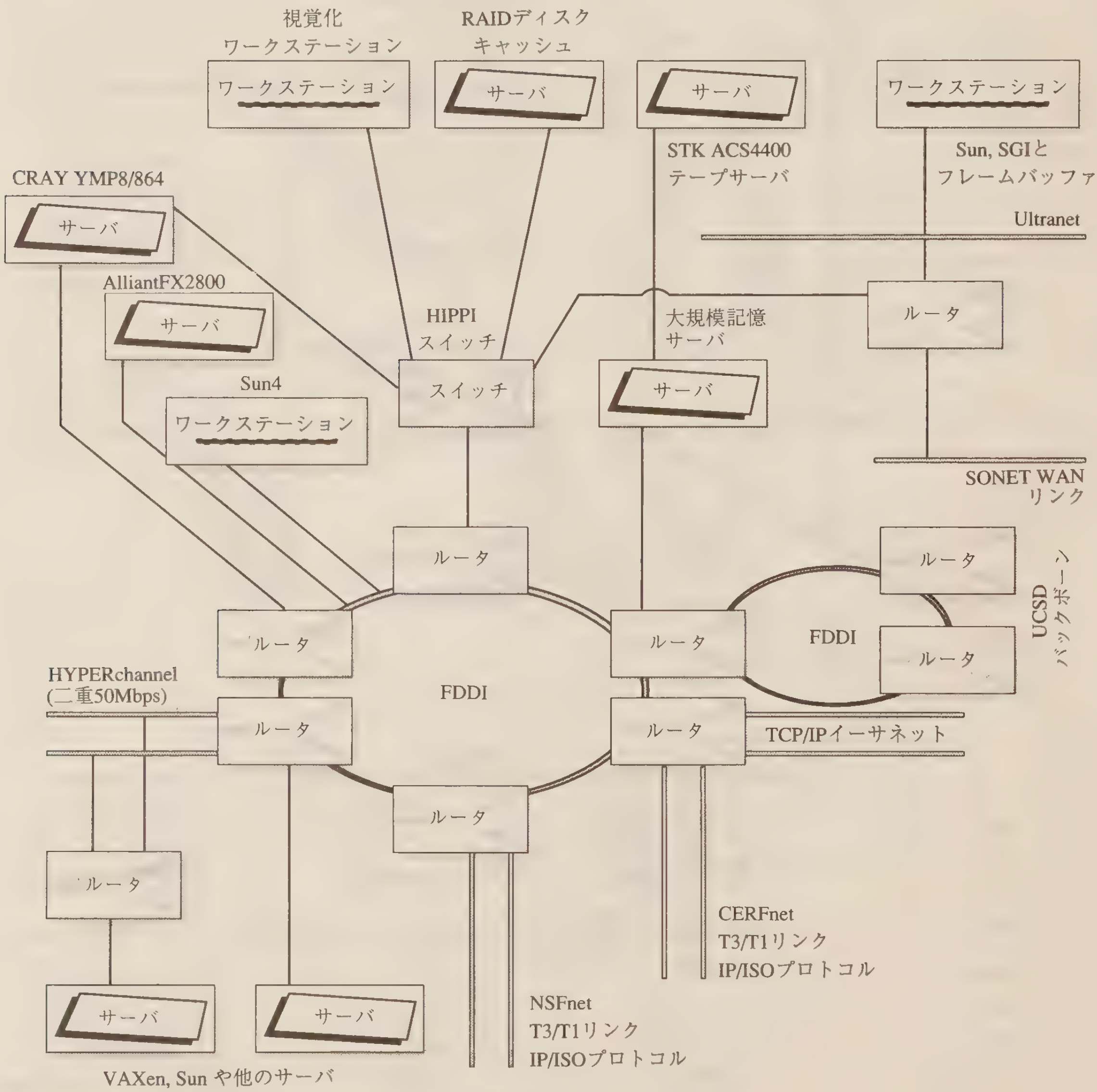
2-23　CERFnet



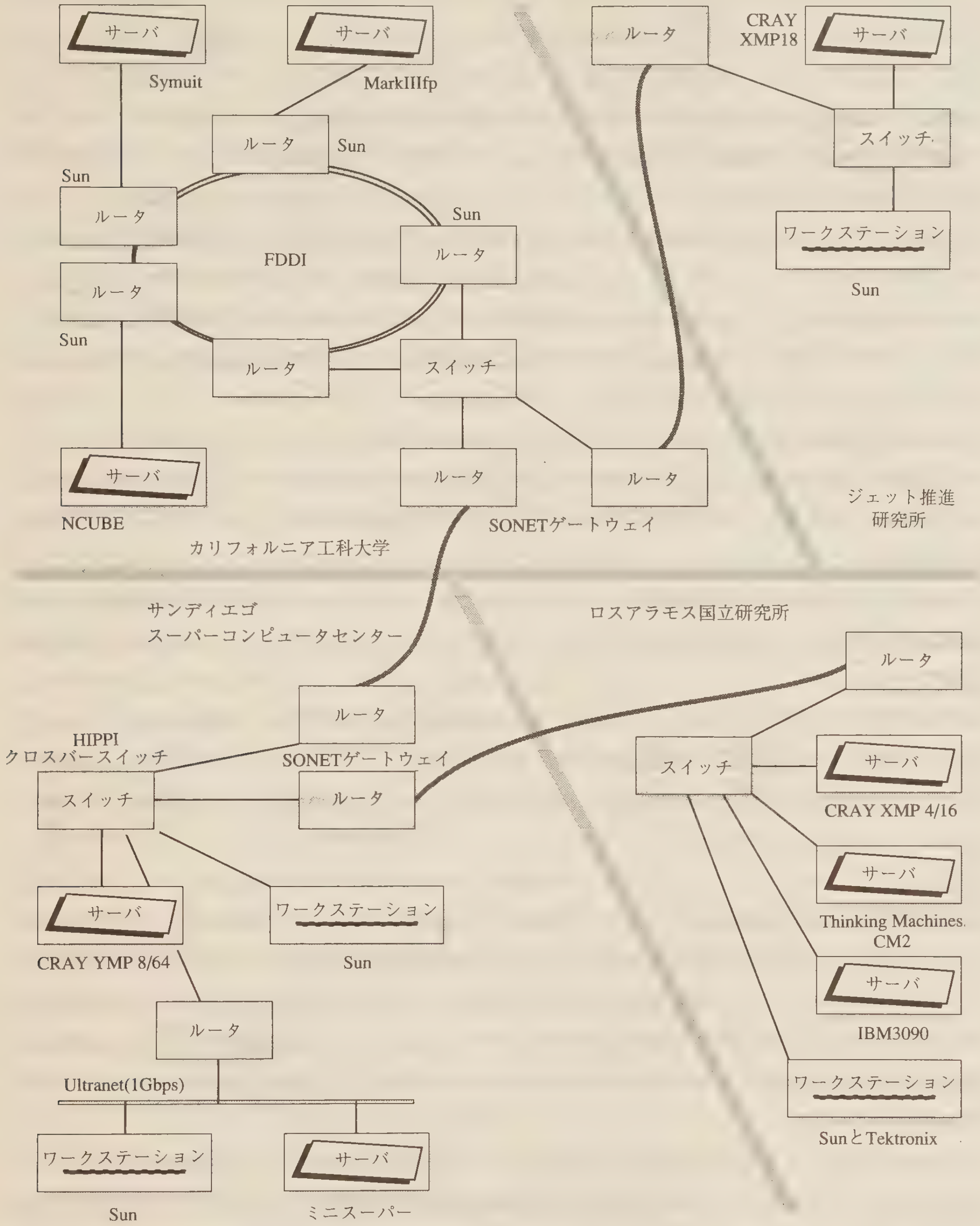
2-24 SDSCネットワーク

Amdahl はさらに，別に 100 ギガバイトのディスク領域をもっている．これに加えて，自動化されたテープシステムが接続されていて，ユーザは 30 秒以内で 3 テラバイト以上のデータにアクセスすることができる．この情報を視覚化したい読者のために付け加えるなら，3 テラバイトというのは 3.5 インチフロッピーディスクを 8 マイル分積み上げたものに相当する．

IBM ファイルサーバと Cray コンピュータは HYPERchannel 経由で通信することもできるが，Ultranet ネットワーク上の別の経路を利用することもできる．この Ultranet ネットワークも，フ



2-25 SDSCネットワークの移行計画



2-26 CASAギガビットネットワーク

レームバッファや Sun ワークステーションなどのシステムをもっている．この高速視覚化ネットワークはファイルサーバとして働く Alliant FX80 ももっている．

このようなネットワークは常に進化している．図 2-25 は SDSC ネットワークの次のフェーズを示している．核となるネットワークが FDDI に基づいたものとなり，1 つは SDSC バックボーンとなり，もう 1 つが SDSC のあるカリフォルニア大学サンディエゴ校のバックボーンとなっていることに気をつけてほしい．

FDDI バックボーンは CERFnet と NSFnet へのリンクをもっており，TCP/IP と OSI の両方のプロトコルをルーティングする．また，DECnet や他の環境への各種の WAN リンクもある．HYPERchannel はここでも用いられており(図の左下方)，Ultranet も用いられている(右上方)．しかし，これらに加えて 800Mbps で動作する高性能並列インターフェイス（High Performance Parallel Interface : HIPPI)に基づいたスイッチもある．

Network Systems 社で作られたこのスイッチは，実際にはクロスバースイッチであり，2 つのノードの間の回線を物理的にセットアップする．このスイッチは数ギガビットもの総バンド幅をもつ．このスイッチには，主サーバシステムや Ultranet や FDDI ネットワークへのルータが接続されている．

HIPPI スイッチはまた，高機能ディスクの冗長アレイ(Redundant Array of Intelligent Disks : RAID)として知られる技術に基づいたディスクキャッシュをもっている．これの基本的な理論は，安くて高速なディスクドライブをたくさんもつ方が，大きなドライブを 1 つだけもつよりもよいということである．小型のドライブは，並列に動作させることによって性能を向上させることができ，データを保護するために冗長なドライブを用意することが許されるほど安い．HIPPI スイッチは大規模並列プロセッサや他のスーパーコンピュータを接続するのにも用いることができる．

図 2-26 は，いかにしてネットワークへの広域アクセスが進化するかを示している．SDSC は，CASA と呼ばれる広域ギガビットのテストベッドの参加者である．このテストベッドの他のサイトは，カリフォルニア工科大学(Caltech)，ジェット推進研究所，ロスアラモス国立研究所である．

システム間のリンクは，長距離会社の提供する光ファイバケーブル上で SONET 標準を用いている．WAN ネットワークへの SONET のゲートウェイは，ローカル HIPPI スイッチに接続されており，今度はこのスイッチが FDDI や HIPPI や Ultranet などの標準に基づいたネットワークへ接続されている．

CASA ネットワークは，これらのスーパーコンピュータサイトのどの 1 つの設備を用いても十分とはいえないような，いくつかの緊急の研究課題を解決する助けとなっている．さらに CASA は，他の 4 つのテストベッドとともに，ギガビット速度で動作する国立研究教育ネットワーク(National Research and Education Network : NREN)に関する経験を提供する助けにもなっている．

2.10 パケット無線

さてここで広域ネットワークのポルシェからフォルクスワーゲンへ乗り換えよう．おそらく，TCP/IP に新しいサブネットワークを追加することがいかに容易であるかを示す最もすばらしい例は，Phil Karn による KA9Q パッケージであろう．このパブリックドメインのプログラムはパーソナルコンピュータに TCP/IP を実現し，ネットワーク中で専用ルータやパーソナルコンピュータクライアントとして用いられている．

しかし，このプログラムの本来の目的はパケット無線をサポートすることであった．データパケットを送るのに無線電波を用いることは，もともとハワイの Aloha ネットワークにおいて行われていた．

パケット無線はまた，アマチュア無線(ham)ネットワークにおいても用いることができる．Phil Karn は，現在世界中でデータを受信することのできる設備をもっている人は 30 000 人にのぼると見積っている．アマチュア無線は厳格にアマチュアである．すなわち，誰もアマチュア無線のサービスに課金をしたり，お金を支払ったりすることはできない．したがって，この環境では低コストの設備が必要とされる．アマチュア無線ではいろいろな伝搬技術が用いられる．例えば，直線見通し，電離層反射，周回衛星，丘の上におかれた反射器などである．月を受動的な反射器として用いることさえある．

パケット無線のモデムは典型的には Terminal Node Controller（TNC）と呼ばれる専用目的の箱に組み込まれている．TNC はまた，パケットプロトコルを実行するファームウェアを備えたシングルボードコンピュータを内蔵している．典型的には 2 つのポートが備えられている．1 つはホストコンピュータとの RS-232 接続のためのものであり，もう 1 つは無線機の音声入出力および送受信切替えスイッチとの接続のためのものである．

Karn は，パケット無線のチャネルは「性質の良い導線やファイバの伝送路とははるかに異なる」と指摘している．パケット無線局は全員が送信することができる点や，(何かしら非決定的な)再送方式が用いられている点ではイーサネットに似ている．

しかし，大きく異なる点もある．まず，イーサネットほどキャリア検出がうまくいかない．すべての局が他のすべての局の送信を傍受することはできない．さらに，送信中の局が受信を同時に行うこともできない（このため衝突検出が難しい)．

しかし，最大の違いは，信号の品質である．バンド幅や出力制限のため，イーサネットに比べてビット誤り率はかなり悪く，信号速度はずっと遅い．この信号品質に対処するため，AX.25 と呼ばれる特別なリンクレベルプロトコルが用いられている．これは X.25 とは異なることに注意してほしい．もっとも，X.25 でも用いられている Link Access Procudere B（LAPB）と呼ばれる HDLC のサブセットを用いている．

AX.25はパケットをHDLCのフレームに入れて同期的に送り出す．アクセスチャネルが共有されているため，典型的には半二重動作が用いられる．ヘッダにはアドレスヘッダがあり，最低限のものとしてFCCにより割り当てられた宛先と送信元のコールサインがASCIIで格納されている．ソースフィールドの後には8つまでの“digipeater”のアドレスがある．これらは，パケットを（通常同一の周波数で）受信，蓄積し，再送信することのできる中間局である．これは厳格な発信元経路指定である．すなわち送信元がdigipeater列全体を指定する．

次に，LAPBの制御フィールドとプロトコルID(PID)バイトが来る．PIDはしばしばF0であり，上位レベルプロトコルは用いられておらず，データを直接端末に送るべきことを表している．このように用いた場合，AX.25は単に2つの端末が“おしゃべり”をするための手段である．

AX.25は，情報（Information : I）フレームと番号なし情報（Unnumbered Information : UI）フレームの2種類のカプセル化方式を用いる．これらはどちらも必要である．Iフレームは明示的に確認され，長距離の短波帯のフェージングや雑音の多い300bps回線で用いられる．UIフレームは超短波帯の見通し56kbps通信路で，Iフレームのオーバヘッドが許容しがたい場合に用いられる．あるモードを用いているときに他のモードを選択するには，IPのサービス種別（Type-of-service : TOS）ビットを用いる．

最初のTCP/IPの実験は，Richard BisbeyとDave Mills（ネットワーク時間プロトコルの創設者)によりなされた．どちらも既存のIPゲートウェイのハードウェアとソフトウェアを改造して用いた．

その後，Karnがアマチュアパケット無線専用のInternetソフトウェアパッケージを創り出した．これがKA9Q Internetプロトコルパッケージである．登録ドメイン（ampr. org）およびクラスAのネットワークアドレス(ネットワーク番号44)もある．これらすべてがアマチュアパケット無線（Amateur Packet Radio）サブネットワーク／ネットワークのためのものである．

典型的なネットワークでは，サブネットワークの考え方ははっきりとしている．すなわち互いに接続されて1つのネットワークを形づくる一連のデータリンクのことである．これに対してアマチュアパケット無線ネットワーク（AMPRnet）は，1つの局と，その場限りの回線から構成されている．通常のネットワークで見られるように，サブネットワークどうしが互いに接続されているわけではない．孤立した島があるだけである．

アドレス空間を割り当てるために用いられている手法は“一般化されたサブネット化”である．すなわち，32ビットのアドレス空間全体がサブネットマスクである．基本的には，ルーティングテーブルの各エントリは固有のサブネットマスクをもっている．ルーティングの検索をするときには，最も広いサブネットマスクをもつエントリを返す．これにより，任意のネットワークトポロジーを実現することが可能になる．ホストに固有なルーティングエントリはすべて1のサブネットマスクである．デフォルトの経路はすべて0のマスクである．

これらすべてがKA9Q Internetプロトコルパッケージにまとめられた．アマチュア無線の世界

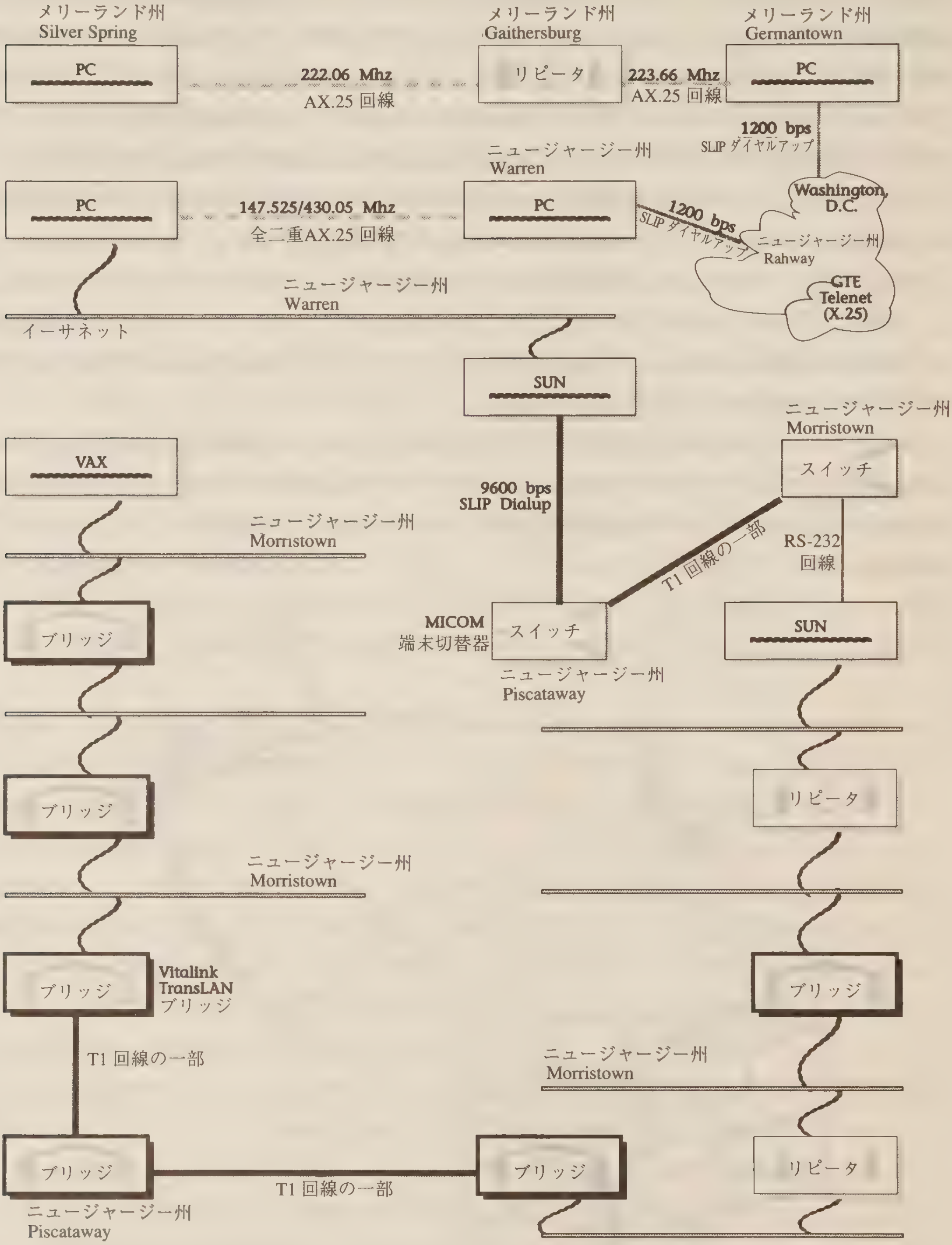
では Sun や VAX はどちらかというと稀なので，このパッケージは当初 IBM PC 用に開発された．基本パッケージは IP，ICMP，TCP，UDP，そして 3 大サービス (Telnet，FTP，SMTP) をサポートしている．また，AX.25 以外に，SLIP，PPP，そして 3Com のイーサネットボードを含む，いろいろなサブネットワークプロトコルがサポートされている．本来の環境は MS-DOS であったが，その後このプログラムは Apple の Macintosh，Commodore の Amiga，Atari の ST，そして Unix System V のいろいろな亜流へと移植されている．当初の版以降，多くのユーザや Phil Karn の大いなる努力によって，このパッケージは非常に強化されてきた．

ひとたびパーソナルコンピュータのシステムが動きだすと，見本となるネットワークが作られた．このネットワークはワシントン D. C. およびその近辺と，ニュージャージーにある設備を用いた（図 2-27 参照）．これら 2 つの地域の間は X.25 のネットワーク（Tymnet）で結ばれ，両側では，様々な回線が用いられた（図 2-28 参照）．

この“アマチュア Internet”は，回線をつなぎ合わせることがいかに簡単であるかを示している．このネットワークはメリーランド州のシルバースプリングに始まり，そこでメリーランド州のジャーマンタウンへのアマチュア無線回線（中間にリピータあり）が開設された．そこから，X.25 回線を経由して，トラフィックはニュージャージー州ワレンにある IBM PC へと送られる．



図 2-27 アマチュアパケット Internet



2-28　KA9Q実験ネットワーク

次いで，ニュージャージーにあるシステムが，パケット無線を用いて，ワレンのバックボーン上を Sun により構成されたルータへとパケットを運ぶ．そこで，SLIP プロトコルにより，端末切替え器へデータが運ばれ，最終的にデータはニュージャージー州モーリスタウンにあるもう 1 台の Sun ルータへと到着する．

モーリスタウンに到着したパケットは，一連のリピータ，ブリッジ，WAN ブリッジ，そしてさらにブリッジを通り抜けて，最終的に VAX サーバに到達する．そこでユーザは何をするのだろうか．それがこの本のこれ以降の主題である．まず，いろいろなデータ回線を一様なネットワークへと統合する TCP/IP プロトコルについて詳しく見る．次いで，これらの回線上で実行することのできるいろいろなサービスを示そう．

より進んだ勉強のために

ANSI, "High-Performance Parallel Interface—Framing Protocol (HIPPI-FP)." X3T9.3/89-146, X3T9.3/89-013, Rev. 3.1, January 23, 1991.

———, "High-Performance Parallel Interface—Encapsulation of ISO 8802-2 (IEEE Std 802.2) Logical Link Control Protocol Data Units (HIPPI-LE)." X3T9.3/119-Rev. 2.0, December 3, 1990.

———, "High-Performance Parallel Interface—Physical Switch Control (HIPPI-SC)." X3T9.3/90-Rev. 1.6, February 28, 1991.

Jacobson, V., "Congestion Control and Avoidance," *Proceedings ACM SIGCOMM '88,* Stanford, Ca., August 1988.

Phil Karn, Amateur Packet Radio and TCP/IP, *ConneXions,* pp. 8-15, v. 2, no. 9, September, 1988.

Lang, Lawrence J., and Watson, James, "Connecting Remote FDDI Installations" *Computer Communication Review,* p. 72, v. 20, no. 3, July 1990.

Malamud, Carl, *STACKS,* Prentice Hall (Englewood Cliffs, NJ: 1991). SMDS, HIPPI, B-ISDN に関して述べられている．

———, *Analyzing Novell Networks.* Van Nostrand Reinhold (New York: 1990). ARCnet およびトークンリングに関して述べられている．

John Romkey, *A Nonstandard for Transmission of IP Datagrams over Serial Lines: SLIP,* RFC 1055, June, 1988.

Ross, Floyd E., et. al., "FDDI: A LAN among MANs" *Computer Communication Review,* p. 16., v. 20, no. 3, July, 1990.

Mark Wolter, "Fiber Distributed Data Interface (FDDI) - A Tutorial," *ConneXions,* pp. 16-26, vol. 4, no. 10, October 1990.

Chapter 3

ネットワーク

3.1 ネットワーク

前章では，個々の部分を形成しているデータリンクの個々の観点からネットワークを見た．それらのものとしては，イーサネット，シリアル回線，X.25，FDDI，その他コンピュータを相互接続するための様々な技術があった．

この章ではひとつ後戻りして，再びネットワークを見てみよう．データリンクは，データパケットをあるノードから別のノードに送るためのものである．ルータは2つの異なるデータリンク間の出入口である．ある種のネットワークアプリケーションは，単一のデータリンク内に閉じているかもしれない．例えば，ワードプロセッサによる作業は，部局内のイーサネット上に置かれたワークステーションとサーバから構成されるかもしれない．しかしたいていは通信しようとする2つのノード――クライアントとサーバ――は，異なるサブネットワーク上に置かれていても構わない．

ネットワーク層は，基礎となるデータリンクサービスの上に構築され，ネットワーク上の任意の2つのノードが相互にアクセスできるようにする．データリンクサービスの場合と同様に，データパケットを転送することだけが問題である．そのパケットの意味や宛先は，上位層が扱うものである．ネットワーク層は，それゆえコネクションレスネットワークサービスであり，最善努力(best-effort)の転送を提供する．これがTCP/IPのIPの部分であるInternetプロトコルが提供する基本サービスである．パケットはルータに送られ，別のルータへと転送され，それがまたいつかは宛先のノードに届けられる．

Sunコンピュータの本来のインターフェイスはIPプロトコルであるが，SunLinkゲートウェイシステムを用いれば，OSIのコネクションレスネットワークサービス(Connection Less Network Service : CLNS)のネットワーク層プロトコルを動作させることもできる．複数のネットワーク(およびトランスポート)プロトコルを統合する問題については，この章の後ろの方で再度触れた後，16章でより詳しく述べる．

ネットワーク層は宛先ノードにデータを届ける(あるいは少なくとも届けようとする)が，よりよいサービスの必要性がなお存在する．それはコンピュータ間ではなく，プログラム間の通信である．あるプログラムやアプリケーションは，同一コンピュータ上に存在する場合もあれば，異なるコンピュータ上の場合もある，他の様々なアプリケーションと通信する可能性がある．

トランスポート層はIP層からのパケットを受け取り，ポートによって識別されるそれぞれのユーザを区別しなければならない．トランスポートプロトコルにも様々なものがある．NFSなどのアプリケーションに用いられているユーザデータグラムプロトコル(User Datagram Protocol : UDP)は，データをプログラムに届けるという基本的なサービスだけを提供している．

伝送制御プロトコル(Transmission Control Protocol : TCP)のような他のトランスポートプロトコルは，より高度なサービスを提供している．TCPはデータを宛先に届けるだけでなく，データが順序通りに到着することを保証する．順序通りに到着することを保証することは，エンド・ツー・エンドのデータ配達を保証することであり，このサービスによってアプリケーションは，送信したメッセージが受信されることを仮定することができる．

トランスポート層とネットワーク層のプロトコルに加えて，様々なサービスプロトコルに触れる．ネットワーク層からデータリンク層のアドレスのマッピングを可能とするアドレス解決プロトコル(ARP)のように本質的に非常に重要なものもある．ルーティングプロトコルのように，もっと複雑なプロトコルもある．ルータはパケットを受け取ってそれを正しい方向に送り出す，と述べたのを覚えているだろうか．大規模なネットワークでは，多くのデータリンクや中継ゲートウェイの中からどれを使うかを決めるのは困難なこともある．

1つあるいはそれ以上のデータリンクの組合せから構成されるネットワークは，ルーティング領域に分割される．ルーティング領域の中では，すべてのルータに対して到達可能な宛先や最適な経路を知らせるようなプロトコルが使われる．経路情報プロトコル(Routing Information Protocol : RIP)はこうしたプロトコルの1つである．

これに対して外部ゲートウェイプロトコル(exterior gateway protocol)のような領域間(interdomain)のルーティングプロトコルもある．これらは，到達可能性に関する情報をルーティング領域をまたいで交換することを可能としている．インターネットは独立に管理されている多くの様々なネットワークから構成されているので，これらの“のりづけ用(glue)”プロトコルは重要である．外部プロトコルにより，独立した管理母体が外の世界に対してどのような接続性を与えるかを正確に決めることができる．

3.2　Internet プロトコル

Internet プロトコルは，ユーザ(TCPやUDPモジュールなど)が次のゲートウェイや宛先のホストにパケットを送ることを可能にする機構である．IPは，アドレッシング(addressing)と断片化

(fragmentation) という2つの基本機能をもっている.

Internet プロトコルは, コネクションレスネットワークサービスであり, これはすなわちそれぞれのデータグラムが独立なエンティティであることを意味する. Internet プロトコルはパケットをネットワーク上の次の場所に転送するが, サービスの保証はしない.

IP はより複雑なサービスを提供するのを助けるために, 以下の4種類の機構を用いる.

・サービスの種別

・余命 (time-to-live)

・オプション

・ヘッダチェックサム

サービスの種別とは単に, 信頼性やスループットやその他の指標に基づいた, 特定の等級のサービスを要求する抽象的な尺度 (metric) である. 実際には, この尺度に基づいて経路を決定するルータはほとんどない. 多くの場合, 単にそれを無視してすべての種別のパケットに対して同じ経路を選ぶのである.

余命 (time-to-live : TTL) の指標はパケットの生存時間の上限である. 経路中のそれぞれの地点においてこの数は減らされる. パケットが宛先に届くよりも前にこの値が0になると, データグラムは破棄される. オプションではパケットごとのサービスを指示することができる. オプションには, タイムスタンプ, セキュリティ, ルーティングに関する特別な指示など様々なものがある.

3.2.1 アドレッシング

名前とアドレスと経路とは区別する必要がある. 典型的には名前づけ (naming) サービスなどの上位レベルが, 論理名から IP アドレスへのマッピングを与える. IP は Internet アドレスをネットワーク内のアドレスに局所的にマップする. ルーティングは IP モジュールの範囲外である. 離れたゲートウェイへの到達法を決めるのは, ルーティングプロトコルが扱う.

アドレスは 32bit の固定長である. アドレスはネットワーク番号とローカルアドレスとから構成されている (図 3-1 参照). IP プロトコルでは異なる数のホストをもつことができるような3つの種類のアドレスが定義されている. クラス A は非常に大規模なネットワークだけに使われ, クラス C は 255 以下のホストをもつネットワークに使われる. クラス B のネットワーク空間は多くの環境にとって最も適切なものであり, アドレス割当て機構はクラス B のアドレスを使い果たしてしまいそうに見える. いつこれが起こるかは, Internet がどれだけ速く成長するかによる.

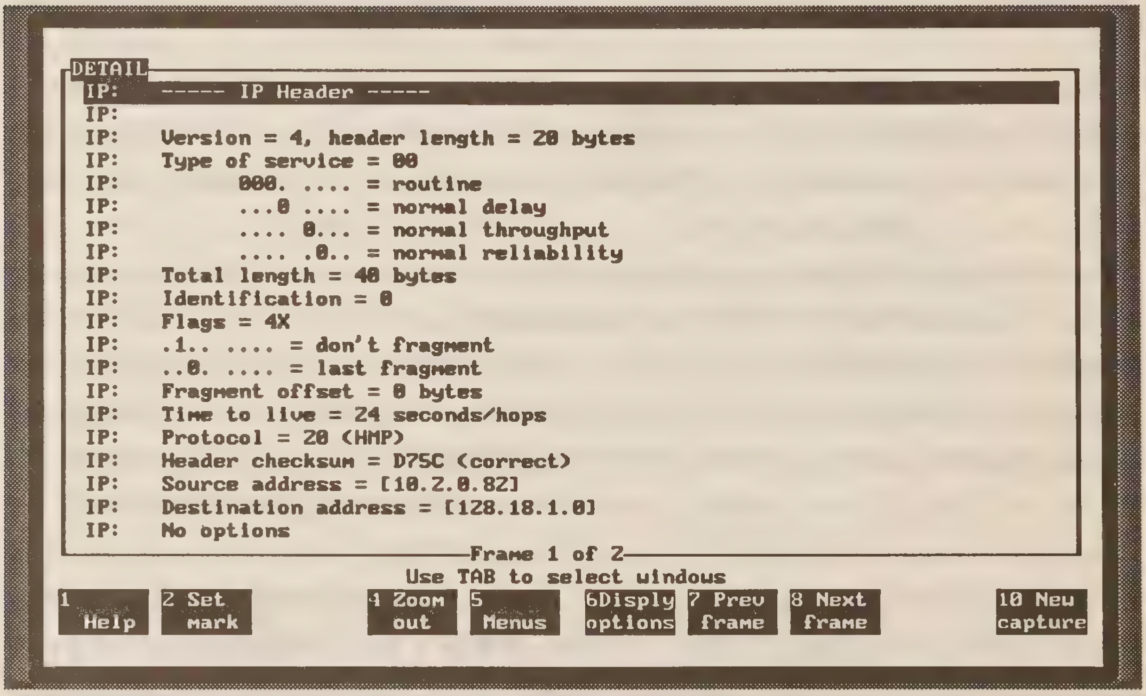
いくつかの Internet のホストはゲートウェイであり, 接続されているネットワークごとに1つずつの, いくつかの異なる Internet アドレスをもっている. いくつかの物理インターフェイスをもつホストもあることにも注意してほしい. このため, 論理 IP アドレスを局所的なネットアドレスにマッピングするときには注意が必要である. 物理インターフェイスのそれぞれに論理 Internet アドレスを与えなければならない.

クラス A	先頭ビット 0
	7ビットのネットワーク番号
	24ビットのローカルアドレス
クラス B	先頭ビット 10
	14ビットのネットワーク番号
	16ビットのローカルアドレス
クラス C	先頭ビット 110
	21ビットのネットワーク番号
	8ビットのローカルアドレス
拡張アドレスモード	先頭ビット 111

3-1 IPアドレスのモード

3.2.2 基本動作

図 3-2 はパケットの IP ヘッダの例を示している．この IP パケットは，宛先のホストまたは中継ルータという他の IP ノードへ届けるために，データリンク層へと渡される．一方，IP 層はといえば TCP トランスポートプロトコルのような上位のユーザのためにデータを運んでいるかもしれない．もし宛先のホストが同じデータリンク上にあれば，IP パケットは直接宛先に転送しろとい



Courtesy of Network General

3-2 IPヘッダ

う指令とともにデータリンク層に渡される．この場合，データリンク層のアドレスは，パケット中のIPアドレスに対応する．

これに対し，宛先のホストが別のネットワーク上にある場合には，パケットはデータリンクサービスを通してある中継ルータへと送られる．そのルータは，IPパケットの宛先アドレスが他のノードであることを見て，パケットがその最終的な宛先に1歩近づくように適当な転送を行う．

IPパケットヘッダの中に余命（TTL）フィールドがあることに注意されたい．どのホストもパケットを受け取ると，TTLフィールドを少なくとも1だけ減少させる．TTLフィールドが0になると，そのパケットは即座に廃棄される．前述したように，IPサービスは保証されたサービスではない．データの配達を保証するのはこの場合トランスポート層であり，ネットワーク層ではない．

パケットにはプロトコルアドレスも含まれており，それはIPサービスの上位レベルのユーザに対応する．IPは，TCPおよびUDPのトランスポート層や，Internet制御メッセージプロトコル(ICMP)のような種々のサポートプロトコルといった様々なモジュールにサービスを提供している．

IPヘッダの最後の部分はオプションのために確保されている(図3-3参照)．オプションは，特定のパケットをネットワーク上でどのように送るかを正確に制御するために使われる．例えば，厳格なルーティング(strict routing)オプションを見てみよう．このオプションには，このパケットを運ぶのに使うべき中継ルータの正確なリストが含まれている．これらのシステムのうち1つでも利用不可能であれば，パケットは廃棄される．

3.2.3 断片化(fragmentation)

パケットに断片化禁止フラグ(don't fragment flag)の印を付けることは可能である．パケットを受け取ったゲートウェイのリンクのMTUが小さすぎると，パケットは断片化(fragment)されずに単に廃棄される．

IPヘッダのIDフィールドは，複数の断片から構成される論理セグメントを識別するために用いられる．断片のオフセットと長さは，セグメント全体の中でのその断片の位置を示している．

ひとたびパケットが途中で断片化されると，最終の宛先に届くまで再度組み立てられることはない．トポロジーが不安定なときやルータが負荷の均一化を行っているときには，それぞれの断片が異なる経路を通っていくかも知れないので，最終の宛先に到着しないうちに再組立の心配をするのは無意味である．

3.2.4 サブネットワーク

管理エンティティがもっていたネットワーク番号の割当て機能を，ネットワークの一部分に関して他の人たちに委託しなければならなくなると，(クラスA，B，Cという)“ネットワーク”の

IPオプションフィールド			
オプションタイプオクテット			3フィールド
		1ビット	コピーフラグ(0=非コピー)
		2ビット	オプションクラス
			0：制御
			1：予約
			2：デバッグ
			3：予約
		5ビット	オプション番号
オプションクラス	番号	長さ	種類
0	0	-	オプションリストの終わり：全1オクテット(長さなし)
0	1	-	何もしない
0	2	11	セキュリティ
0	3	可変	厳格でない発信元ルーティング
0	9	可変	厳格な発信元ルーティング
0	7	可変	レコード経路
0	8	4	ストリームID
2	4	可変	インターネットタイムスタンプ
セキュリティフィールド			このオプションは断片にコピーされなければならない．1つのデータグラムの中にこのオプションのコピーが1つだけでもよい．
セキュリティフィールド		16ビット	16レベルのセキュリティの1つを記述する．16のうち8つは将来の使用に予約されている．“全ビットが0”は分類されていない．その他は秘密，制限，機密，最高機密である．
区切り		16ビット	セキュリティの区切り．
制限の扱い		16ビット	
伝送制御コード		24ビット	トラフィックを分離し，関心の対象となる制御集団を定義するために使用される．
厳密でない発信元およびレコード経路オプション(LSRR)			
オプションコード			タイプ=131
長さオクテット			
ポインタオクテット			経路データは一連のアドレスである．もしポインタ>長さならば，発信元経路は空(もしくは記録された経路がいっぱい)で，ルーティングは宛先に基づいていなければならない．フィールドが増加するたびに，発信元経路はレコード経路に置き換えられる．厳密でない経路は，ゲートウェイが経路中の次のアドレスに到達するために他のゲートウェイを使うことができるということを意味する．
経路データ			アドレスのリスト．

3-3　IPオプションフィールド

厳密な発信元およびレコード経路		リスト中に記述されていない中継ゲートウェイは使用できない.
オプションコード		タイプ=137
レコード経路		発信元のホストは，経路情報を保つために十分大きなデータブロックと共にこのオプションを構成しなければならない．領域がいっぱいなら，データを挿入することなしに転送を続ける.
オプションコード		タイプ=7
タイムスタンプ		
オプションコード		タイプ=68
オプション長		オプション長の最大値は40である.
ポインタ		重ね書きされた領域の終端へのポインタ．合法な最小値は5.
オーバフローフラグ	4ビット	空間の欠如のためにタイムスタンプを記録できないIPモジュールの番号.
フラグ	4ビット	0：タイムスタンプのみ
		1：インターネットアドレスの後に続くタイムスタンプ
		3：インターネットアドレスフィールドがあらかじめ記述され自分のアドレスがリスト中の次のアドレスに一致する場合，自分のタイムスタンプだけを記録する.
タイムスタンプ	32ビット	世界標準時(UT)真夜中からミリ秒単位で右づめにされる．もし異なるタイムスタンプを用いる場合は，非標準値であることを示すために最上位ビットを1にする

3-3（続き）　IPオプションフィールド

概念はあっという間に路傍に落ちた.

外の世界に対しては，ネットワーク全体が，1つあるいは複数のゲートウェイによってサービスされる単一のネットワークと変わりない．しかし，ひとたびデータがネットワークの中に入ると，ゲートウェイを用いて適切な宛先サブネットワークまで中継されなければならない.

このように1つのネットワークはいくつかのサブネットワークの集まりである．ゲートウェイはちょうどネットワークを相互に接続するのと同様，サブネットワークを相互に接続する．実際にはルーティングに関する限り，サブネットワークとネットワークの違いはない．すなわち，ある宛先に1歩近づくためにはどのゲートウェイが最適かという判断がなされなければならない.

サブネット化の問題を扱うために，アドレスのホスト部からいくつかのビットが盗まれる．例えば，クラスBのアドレスは，本来は14ビットのネットワーク番号(組織に割り当てられる)と16ビットのホストアドレス空間から構成される.

16ビットのアドレス空間をもつ代わりに，8ビットをサブネットワークアドレスに，残りの8ビットをホストの空間に割り当ててもよい．このように，それぞれが255の異なるホストをもつ255までのサブネットワークを作ることができる．普通，0は“この”エンティティを意味する．このネットワーク，このサブネットワーク，このホストという具合いである.

ある組織はクラスBのネットワークアドレスを申請することができる．そしてサブネット化手順を用いて，この空間からクラスCと同等の255のサブネットワークを作ることができる．外の世界に対しては，これは単一のネットワークと等価である．ひとたび管理領域の内側に入ると，適切なサブネットワークに到達するためのさらなるルーティングが行われる．

ホストがパケットをどこに送るかを決定するとき，同一サブネットワーク内か否かが基本的な問題となる．そのパケットが同じサブネットワーク上のホストに行くものであれば，ホストはそのパケットを自分で送ることができる．この場合，IPアドレスをデータリンクアドレスに変換するためにARPパケットを送り，それからパケット本体を送る．

パケットが同一サブネットワーク内に対するものでない場合，パケットはゲートウェイに送られ，そのゲートウェイがそのパケットをしかるべきところに送る．あるアドレスが同一サブネットワーク内か否かをホストが判断する際，32ビット長のマスク，ホスト自身のIPアドレス，宛先のIPアドレスを使用する．サブネットワークマスクは可変長マスクであり，可変長サブネットワークを許している．

基本的な手続きは，アドレスのうちサブネットワークマスクが1にセットされている領域を見て，その値を意味あるものとして扱う．例えばマスクの最初の8ビットが1であって，アドレスの上位8ビットが同じであれば，宛先アドレスはローカルホストと同じサブネットワーク上にある．

では，ホストはどのようにしてそのサブネットワークマスクを見い出すのだろうか．これにはいく通りかのやり方がある．1つは，ICMPアドレスマスク要求およびアドレスマスク応答である．ブートパラメータ(BOOT Parameter : BOOTP)プロトコルにも，サブネットワークマスクのための用意がなされている．

3.3 ISO CLNSとTCPのIP

IPプロトコルと，OSI環境で使われるコネクションレスネットワークプロトコル(CLNP)は，主要な2つのネットワーク層である．Sunワークステーション上ではこの両方がサポートされている．IPはもとからサポートされているプロトコルであるが，CLNPはSunLink製品として付け加えられた．

ウィスコンシン大学のRob Hagensは，2つのプロトコルの比較に関する文献を発表し，サービスインターフェイスに関してはこれら2つはおおよそ同じ機能をもっていることを示した．ユーザには相違は見えないが，内部では確かに違いがある．

最も明白な相違の1つにアドレス空間の大きさがある．IPは32ビットの固定長アドレスを用いるが，ISOは20バイト(160ビット)までの可変長アドレスを用いている．これはまず大きさにおける違いである．

Internetは現在クラスBのアドレスを割り当てており，これらは中規模な大きさのネットワー

クに用いられている．言い換えると，32 ビットのアドレス空間の最も望ましい部分はいっぱいになりつつある．ISO の世界では，よほど浪費的な割当てがなされない限り，20 バイトのアドレス空間がすぐに使い尽くされてしまうことはないであろう．その一方で，1 バイトの Telnet データを運ぶには，20 バイトというのはかなり余分なお荷物である．

アドレッシングの違いとしてはその他にアドレスの形式とアドレスの位置がある．IP では，ヘッダの最大長は 60 バイトであるが，OSI では 255 バイトである．この理由は，ISO は様々な種類のオプションをよく使用するからである．オプションの使用は，良いこともあれば悪いこともある．例えば，CLNP の断片情報はオプションである．断片化されていないパケットでは 6 バイト短くなり得る．CLNP では，すべてのオプションは分断化されたパケットにコピーされる．IP では，それぞれのオプションには，分断化されたパケットにコピーする必要があるかどうかを示すフラグがある．これらのプロトコルは両方とも，似た種類のパラメータをサポートしている．例えば，両方ともセキュリティパラメータをもっている．

また，どちらのプロトコルも，厳格および厳格でない発信元ルーティングをサポートしている．IP では，宛先アドレスフィールドが実際に書き換えられる（そして発信元ルーティングリストが参照されたときに再び書き換えられる）．CLNP では，宛先アドレスはそのままにしておかれる（ただし中継アドレスは記入される）．

両方とも，経路記録オプションをもっている．IP では，アドレスは通過した順にリストされ，帰り道を見つけるためには逆順にしなければならない．CLNP では経路は逆順に記録される．IP では，発信元経路が用いられれば暗黙的に経路記録が得られる．発信元経路アドレスがリストから取り除かれると，いま通っているホストのアドレスに置き換えられる（そのフィールドで利用できる長さまで達すると止まる）．

サービス品質（QOS）の扱いは異なる．IP では，サービスタイプ（TOS）はヘッダの一部で，優先度（precedence），遅延，スループット，信頼性のフラグが含まれている．CLNP では QOS パラメータはオプションの中に含まれる．3 種類の QOS 指示は，発信元アドレス固有情報（意味は未定義），宛先アドレス固有情報（意味は未定義），全域的一意情報である．このように CLNP の方が QOS の定義においては柔軟であるが，適切なものを見い出して用いるのは難しくなっている．

CLNP には，輻輳経験ビットのような全域的一意情報がある．IP にはこれと同等なものはないが，ICMP 発信元抑制（source quench）メッセージがある．IP には IP TOS の中に優先度パラメータがあることに注意されたい．CLNP ではこれは別の優先オプションで行われる．

誤りに関しては，IP では ICMP メッセージが使われるが，CLNP では異なる種類のパケットを用いる内部誤り報告機能が用いられる．どちらの機構も以下のような誤りを報告することができる．

・宛先に到達できない
・TTL が 0 になった

・構文誤り／不適切パラメータ

CLNP には再方向づけ(redirect)のようなことに対する用意はない．ISO の世界では，これらすべての機能は独立したプロトコルである ES-IS(ISO 情報標準 9542)によって扱われる．ICMP 発信元抑制，エコー，タイムスタンプ，情報要求に対しては，ISO のネットワーク層で直接対応するものはない．

3.4 サポートプロトコル：ARP と ICMP

個々のネットワークノードは少なくとも 2 つのアドレスをもっている．1 つは 48 ビットのイーサネットアドレスのようなデータリンクアドレスであり，もう 1 つはネットワークおよびホスト番号からなる 32 ビットの Internet アドレスである．

IP 層は IP アドレスに基づいてルーティングの判断をする．ルーティングテーブルは，例えばデフォルトの経路に用いられる中継ルータの IP アドレスを示している．未知のホスト行きのパケットはこのルータに送られる．

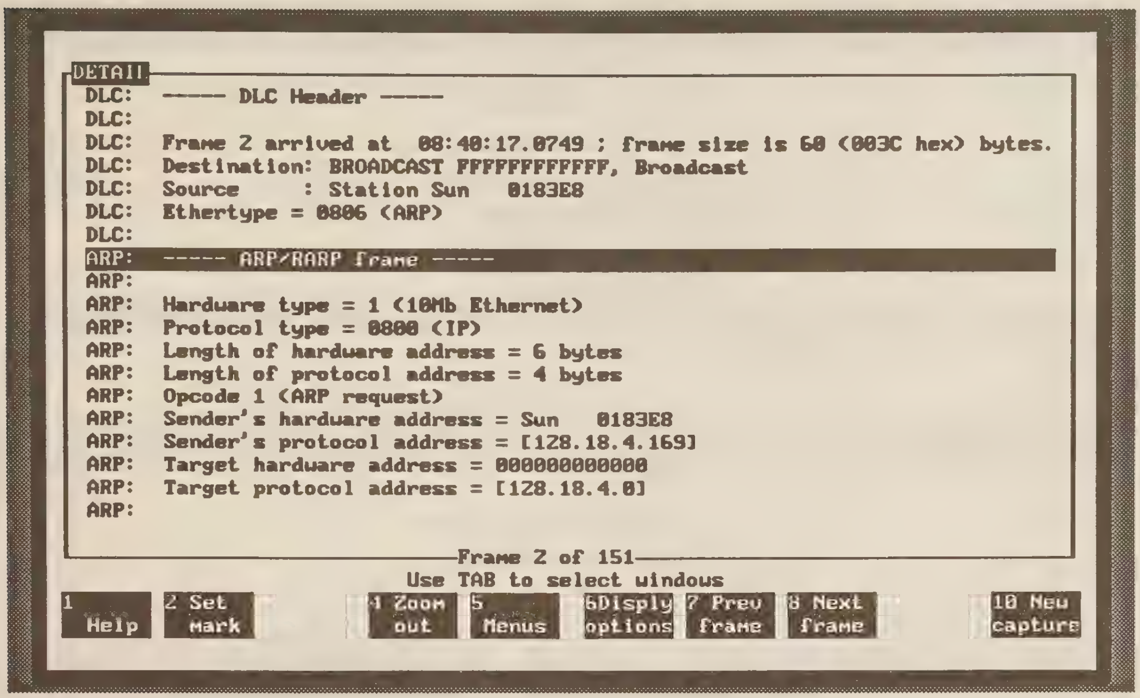
ここで問題となるのはデフォルトのルータの Internet アドレスしか知られていないということである．パケットをこのホストに送るためには，データリンク層に対して必要とするデータリンクアドレスを与えなければならない．アドレス解決プロトコル(ARP)は，この問題を解決してくれる．

SMDS，FDDI，イーサネットのような，IP がサポートしているすべてのマルチアクセスデータリンク層において ARP が存在する．図 3-4 と 3-5 は ARP 要求と ARP 応答の例を示している．この場合，送信側は自分自身のアドレスを知っており，要求の中にその情報を含んでいる．これにより宛先ノードは，IP 応答に必要となる情報をキャッシュすることができる．

ARP パケットの中においては，宛先のインターネットアドレスはわかっており，データリンクのハードウェアアドレスを見い出そうとしている．逆アドレス解決プロトコル(RARP)は似たような問題であり，この場合ホストはハードウェアアドレスは知っているが IP アドレスは知らない．このような状況はディスクなしのノードが起動するときに起こる．ハードウェアアドレスに関する情報は LAN アダプタ ROM の中に設定されているのでわかる．IP と TCP が起動するためには IP アドレスが必要となり，そうすればオペレーティングシステムを転送してシステムを立ち上げることができる．

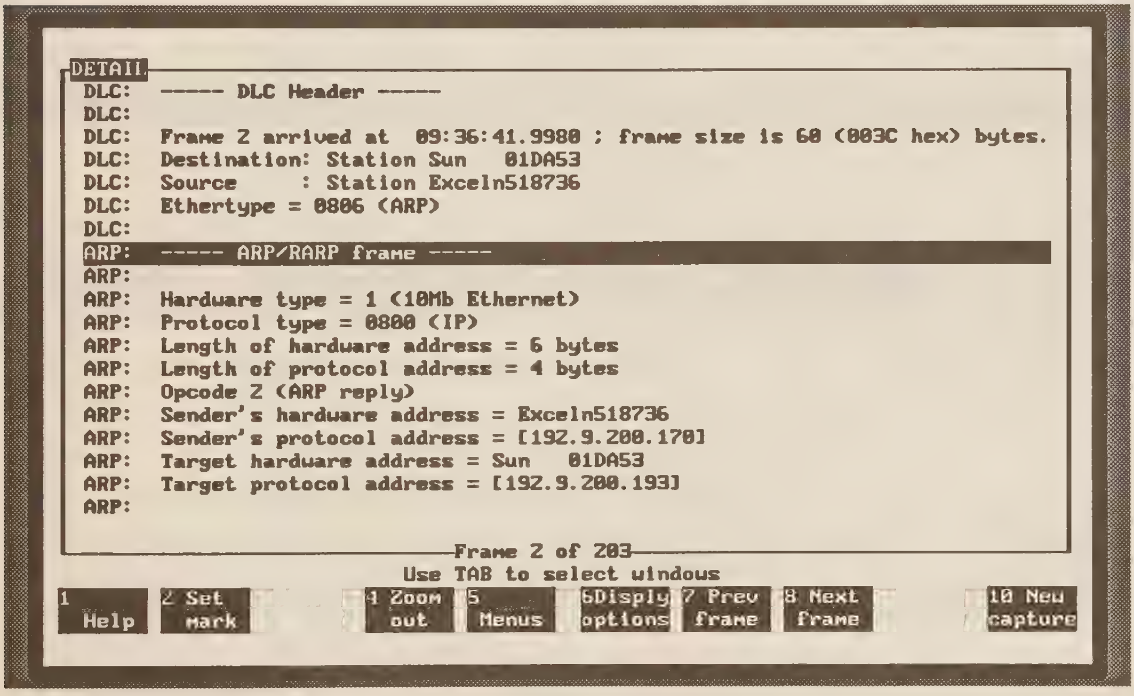
3.5 ICMP

Internet 制御メッセージプロトコル(Internet Control Message Protocol : ICMP)は Internet プロトコルの上に位置するサービスである．それは独立したモジュールであるが，IP とは一体化され



Courtesy of Network General

3-4 ARP/RARPフレーム



Courtesy of Network General

3-5 ARP応答

宛先到達不可能メッセージ		
タイプ	1バイト	タイプ=3
コード	1バイト	0：ネット到達不可能 1：ホスト到達不可能 2：プロトコル到達不可能 3：ポート到達不可能 4：断片化が必要しかしDFセット 5：発信元経路失敗
チェックサム	2バイト	
未使用	32ビット	
本来のデータ		インターネットヘッダ＋本来のデータの64ビット
時間超過メッセージ		
タイプ		タイプ=11
コード		0= 通過中に余命が0となった 1= 断片再組立時間超過
チェックサム		
本来のデータ		インターネットヘッダ＋本来のデータの64ビット
Parameter Problemメッセージ		
タイプ		タイプ=12
コード		0=ポインタが誤りを指示
チェックサム		
ポインタ		コードが0の場合
本来のデータ		インターネットヘッダ＋本来のデータの64ビット
発信元消失メッセージ		
タイプ		タイプ=4
コード		コード=0
チェックサム		
本来のデータ		インターネットヘッダ＋本来のデータの64ビット
再方向づけメッセージ		
タイプ		タイプ=5
コード		0：ネットワークのためのデータグラムの再方向づけ 1：ホストのため 2：サービスのタイプとネットワークのため 3：サービスのタイプとホストのため
チェックサム		
アドレス	32ビット	ゲートウェイインターネットアドレス
本来のデータ		インターネットヘッダ＋本来のデータの64ビット

3-6 ICMPメッセージ

エコーまたはエコー応答メッセージ		
タイプ		エコーの場合8
コード		コード=0
チェックサム		
識別子	2バイト	エコーと応答を一致させるためのもの，0もあり得る
順序番号		さらなる一致情報．典型的IDはTCP/UDPポートで，順序番号はエコー要求が送られるたびに増加させてもよい
データ		エコーバックのために使用
タイムスタンプまたはタイムスタンプ応答メッセージ		
タイプ		タイムスタンプの場合13，応答の場合14
コード		コード=0
チェックサム		
識別子	2バイト	
順序番号	2バイト	
タイムスタンプ開始	32ビット	発信者がメッセージに触った最後の時間
タイムスタンプ受信	32ビット	エコー者が受信時に触った最初の時間
タイムスタンプ伝送	32ビット	エコー者が送信時に触った最後の時間
情報要求または応答メッセージ		0（“この”ネットワークを意味する)にされた発信元と宛先のアドレスが共に送られる．答はネットワーク番号の中に置かれる．ホストはそれらがどこにあるか見つける方法．
タイプ		情報の場合15，応答の場合16
コード		コード=0
チェックサム		
識別子		
順序番号		

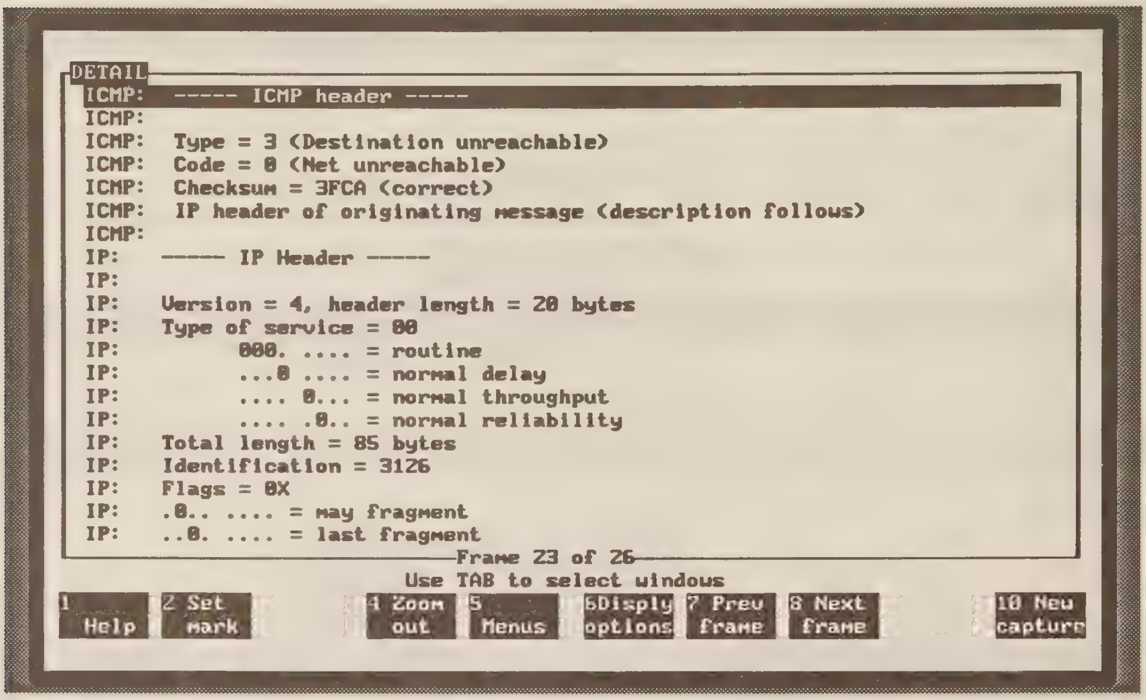
3-6（続き） ICMPメッセージ

た部分であり，すべての実装に含まれなければならない．もちろん，これをサポートしない非常に原始的な動作もいくつかある．BOOTP を用いたディスクレス起動や，自明ファイル転送プロトコル(Trivial File Transmisson Protocol : TFTP)サービスなどは著名な例である．

前述した通り，IP プロトコルではサービスの保証は行わない．パケットは最善努力の原則に基づいて送られる．ICMP は発信ホストにいくらかのフィードバックを提供するが，配達機構として IP を用いているので，このフィードバックもまた最善努力原則に基づく．

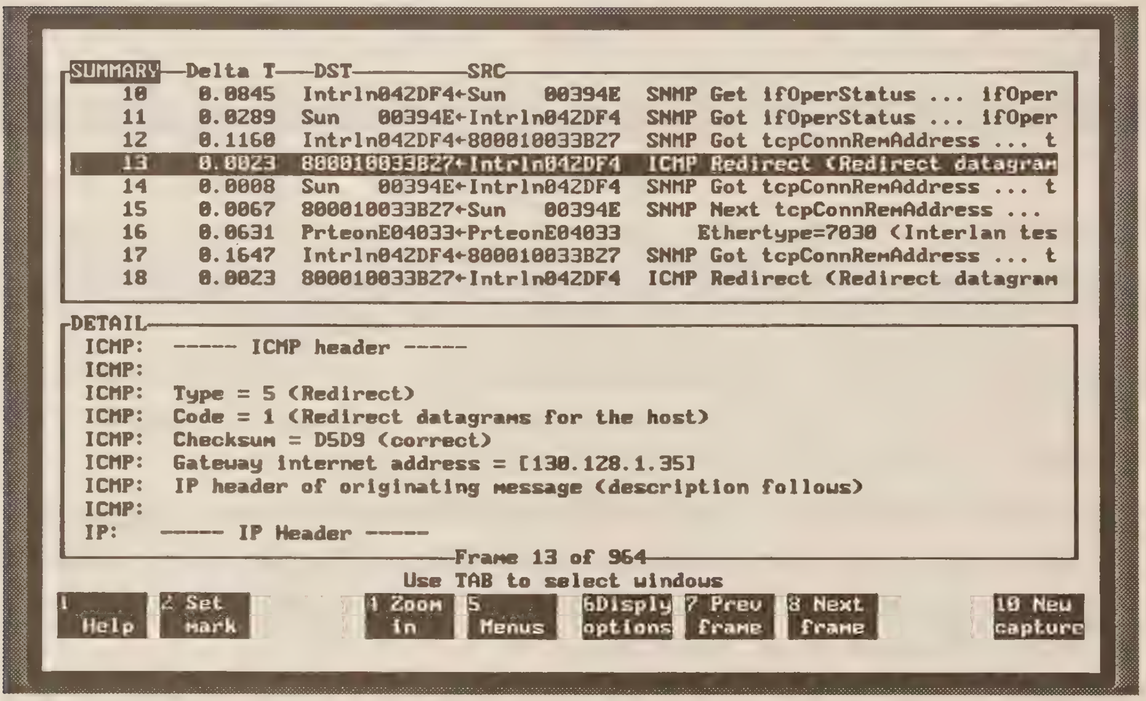
誤りが生じたときや発信ホストに他の忠告を与える必要があるとき，ICMP メッセージが送られる(図 3-6 参照)．ICMP メッセージがループになるのを避けるため，ICMP メッセージに対して ICMP メッセージが送られることはない．断片化されたデータグラムに関しては，断片 0 に関する誤りに対してだけ ICMP メッセージが送られる．

ICMP には 3 種類のメッセージがある．宛先到達不可能は，ルータがパケットをその宛先に送る(あるいは少なくとも宛先に 1 歩近づく)ことができないときに送り返される．このメッセージは



Courtesy of Network General

3-7 ICMP宛先到達不可能



Courtesy of Network General

3-8 ICMP再方向づけメッセージ

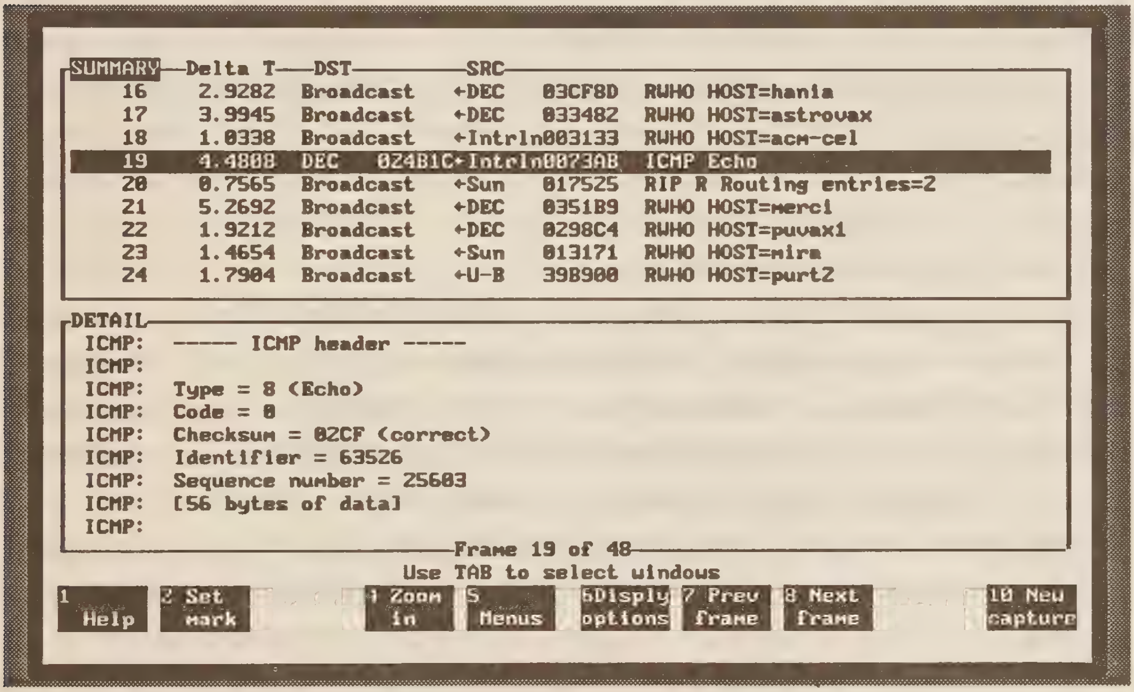
問題となるデータグラムの発信元アドレスに送り返される(図 3-7 参照).

時間超過およびパラメータ不適切メッセージは，IP がパケットのルーティング方法を知っているがルーティングできないときに使用される．余命(TTL)が 0 になると時間超過メッセージが送られる．パケット形式に関して何らかの問題があるとパラメータ不適切報告が送られる．

発信元抑制および再方向づけメッセージの 2 つのメッセージは，ホストに対して現状のネットワークの状態に関するフィードバックを提供する．ルータが能力の限界に達しそうになると，発信元抑制メッセージが送り返され，発信者を引き下がらせようとする．発信者が引き下がるという保証はないが，多くの TCP の実装では，このメッセージに応じて送信レートを調整することができる．

再方向づけメッセージは，中継のためにホストがメッセージをゲートウェイに送ったときに起こる．そのゲートウェイはルーティングテーブルを見て，他のゲートウェイがよりよい仕事ができること，またその第 2 のゲートウェイが発信ホストと(同じサブネットワーク上で)直接アクセス可能であることを知る．再方向づけメッセージを送ることで，発信ホストは将来宛先に到達しようとするときには第 2 のゲートウェイを使うべきであることを知る．

元のパケットは 2 番目のゲートウェイに直接転送される．ICMP が忠告的であることを思い出そう．もし発信ホストが再方向づけを無視したければ，最初のゲートウェイに送り続けるだろう．おそらくこのような貧弱な実装でも動作するだろうが，標準規則の多くに違反するだろう．



Courtesy of Network General

図 3-9 ICMPエコー

その他いくつかの種類のメッセージもある．例えば，エコーおよびエコー応答は，宛先が利用可能かどうかを試験するために ping コマンドによって使用される(図 3-9 参照)．タイムスタンプメッセージはホストの現在のクロックの読みを見い出す方法である．もっとも，後の方でネットワーク時間プロトコルの方がこの問題を扱うにはずっと洗練された方法であることを見ることになるが．

3.6 TCP

伝送制御プロトコル(TCP)は IP の上位層であり，コネクション指向で，エンド・ツー・エンドの信頼性のあるプロトコルである．IP が最終の宛先にデータを届けると TCP が後を引き継ぎ，すべてのデータが届いたことおよび正しい順序であることを確かめる．

TCP のユーザ――アプリケーション――に対する基本サービスは，それぞれの方向に対する連続したオクテットの列（ストリーム）である．ユーザは，どのようにしてデータが宛先に届くか(これは IP の役割)とか，最大転送単位(MTU)といったことに関して心配することはない．

基本 TCP のサービスは，IP モジュールからのパケットを受け取る．複数の異なる TCP のユーザが存在し得る．そこで TCP は到来するパケットを受け取ると，2 つのノードのユーザの対を用いてそれらのパケットをそれぞれのコネクションに割り当てる．TCP へのアクセス点はポートとして知られる．

TCP はひとたびパケットを適切なポートに割り当てると，データの順序が正しいかどうかを調べるために順序番号をチェックする．これに加えて，データを受け取った後で，TCP は確認応答(Acknowledgement)を生成して，データが届いたことを相手ノードに知らせる．到着したデータを再び組み立てることに加えて，TCP モジュールは送信した情報の追尾を行っている．確認応答は送り出したデータと関係がある．すなわち送り出したデータが確認されれば，TCP モジュールはそのデータを廃棄することができる．

もし，何らかの理由で，あるデータが確認されない場合には，発信側の TCP モジュールは確認されていないデータ全部を再送する．相手側が否定的確認を生成することもある．この場合にも再送を行わなくてはならない．確認されずに残っているデータの量はウィンドウとして知られている．ウィンドウはそれぞれの方向ごとに異なっていてもよい．受信側がウィンドウを提示し，これによって送信側は一定の量のデータを送ることが認められる．

送信側はウィンドウも追尾しているが，それは受信側から提示されたウィンドウから，確認されていない送信済みデータの数を差し引いたものである．これは利用可能ウィンドウとして知られている．

コネクションが確立されると，コネクションの同期(SYN)をとるために TCP ヘッダの 1 ビット(図 3-10 参照)がセットされる．このフラグに付け加えて，ノードは初期順序番号(ISN)を送信

発信元ポート	2バイト	
宛先ポート	2バイト	
順序番号	4バイト	SYNがフラグフィールドに存在していなければ，これはデータフィールド中の最初のデータのオクテット番号である．SYN(同期順序番号)がセットされていれば，これは初期順序番号(ISN)で，最初のデータのオクテット番号はISN+1となる．
ACK番号	4バイト	ACKビットがセット(有効ACKフィールド)されていれば，これは次に期待されるバイト値である．これはコネクションが確立さたあと常にセットされる．
フラグ	全64ビット	
データオフセット	4ビット	ヘッダ中の32ビットワードの番号
予約	6ビット	
制御ビット	6ビット	URG：有効緊急ポインタフィールド
		ACK：有効ACKフィールド
		PSH：プッシュ
		RST：リセットコネクション
		SYN：同期順序番号
		FIN：送信側からのデータなし
ウィンドウ	16ビット	この受信側が受け入れようとする，これまでACKされたオクテット数．
チェックサム	16ビット	
緊急ポインタ	16ビット	このセグメント内のオフセット値．緊急データのすぐ後のオクテットを指す．
オプション	可変長	
パディング	可変長	ヘッダを32ビットの整数倍にするためのもの
データ		

3-10 TCPヘッダフォーマット

する. ISN はデータストリームの始まりである. このコネクションで送られるバイトはすべて ISN からのオフセットで表される. 初期順序番号が 1000 で 2000 バイトが送られたなら, 次の順序番号は 3001 である.

全二重コネクションでは両方が ISN を受け取る．この場合は両側とも，送信側であると同時に受信側でもある．データが送られるときには順序番号が含まれており，これによって受信側はデータストリームを再構成し，欠けている部分がないかを検出することができる．

TCP モジュールがデータを送るとき, 確認応答(ACK)を含めることができる. 確認応答はそのモジュールが受け取った最大の順序番号である．送信するパケットに ACK をピギーバックすることにより帯域幅を節約できる. 必要であれば, ACK はデータのないパケットで送ることもできるが，その場合 ACK は TCP(と IP とイーサネット)ヘッダ全部を必要とするのでオーバヘッドは当然大きくなる．

これらの順序番号を管理するためには，いろいろな情報を追尾しておくことが必要である．送信側ノードは以下の4つの値を追尾する必要がある．

- 確認されている旧順序番号
- まだ確認されていない旧順序番号
- 次のデータ転送に用いる順序番号
- まだ使用してはいけない将来の順序番号

最後の番号により，周回(wrap-around)問題が解決される．ISNはなるべく乱数となるように選ばれる．特に長いコネクションでは，ある時点で順序番号が全ビット1になり，番号は周回することになる．

コネクションが非常に高速である場合，周回はかなり早く起こることがある．実際，非常に高速で非常に長距離なデータリンク上では(あるいはこうしたデータリンクを通るあらゆる経路上では)，送信順序番号が未承認データの番号に追いつくこともあり得る．

初期順序番号を乱数で選ぶのは，コネクションを何度も再利用することに関して何の制限もないからである．コネクションとは単にソケットやストリームヘッドの対，すなわちネットワークへの2種類のエントリポイントにすぎないのである．

多くのサイトでは，初期順序番号としてシステムのクロックを用いている．32ビットの順序番号が4μsごとに増加していくと，この32ビットの順序番号は4.55時間で1周することになる．この値はパケットが順序番号が1周するまで生存するのを防ぐには通常十分に大きい値である．

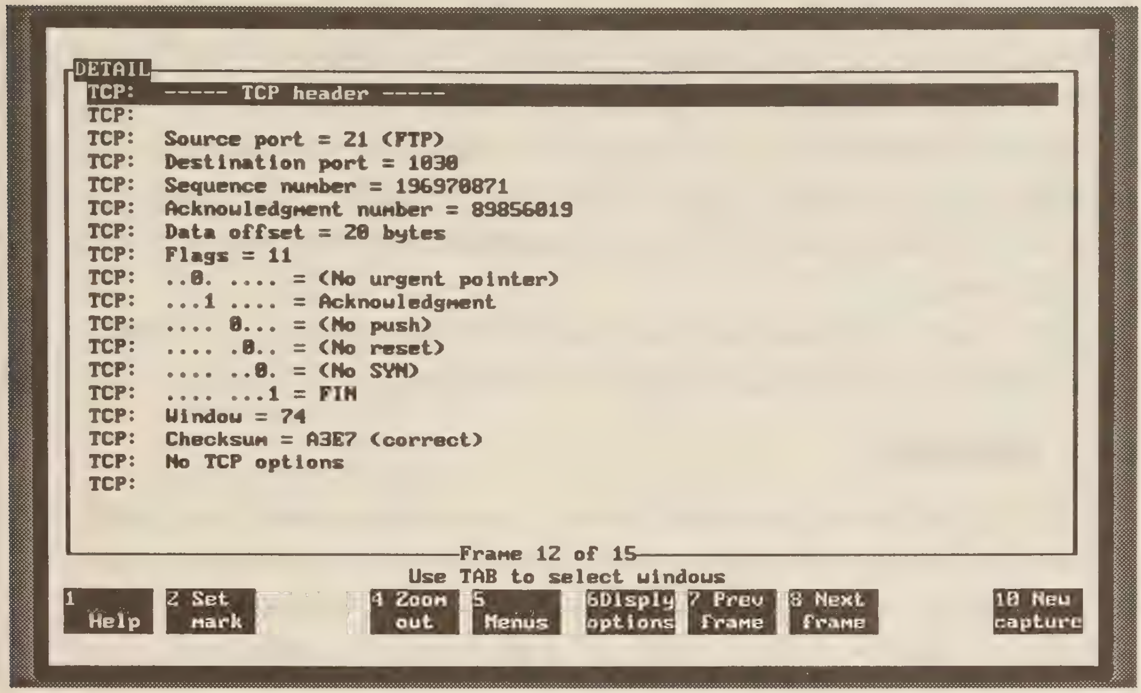
コネクションの確立はTCPではかなり簡単である．コネクション確立の段階では3つのTCPパケットが必要である．まず，設定側のノードによりSYNビットが送られ，順序番号が送られる．すると他方のノードはその順序番号を承認し，自分の順序番号を送る．最後に最初のノードがこの相手のノードの順序番号を承認する．この最初のハンドシェークに従って両方のノードは同期がとられ，次の処理に進むことができる．

図3-11は，IPパケットの中に埋め込まれる典型的なTCPヘッダを示している．そのIPパケットはといえばデータリンクパケット(この場合はイーサネットである)に埋め込まれるわけである．発信元ポートはファイル転送プロトコル(FTP)である．このパケットの宛先は未知のポートであり，ほぼ間違いなくFTPクライアントであろう．

フラグを見れば，このパケットは単なる確認パケットで，利用者データをもっていないことがわかる．確認を与えることに加えて，このパケットは順序番号をもっている．相手ノードはこの順序番号を用いて自分の確認を送ることができる．

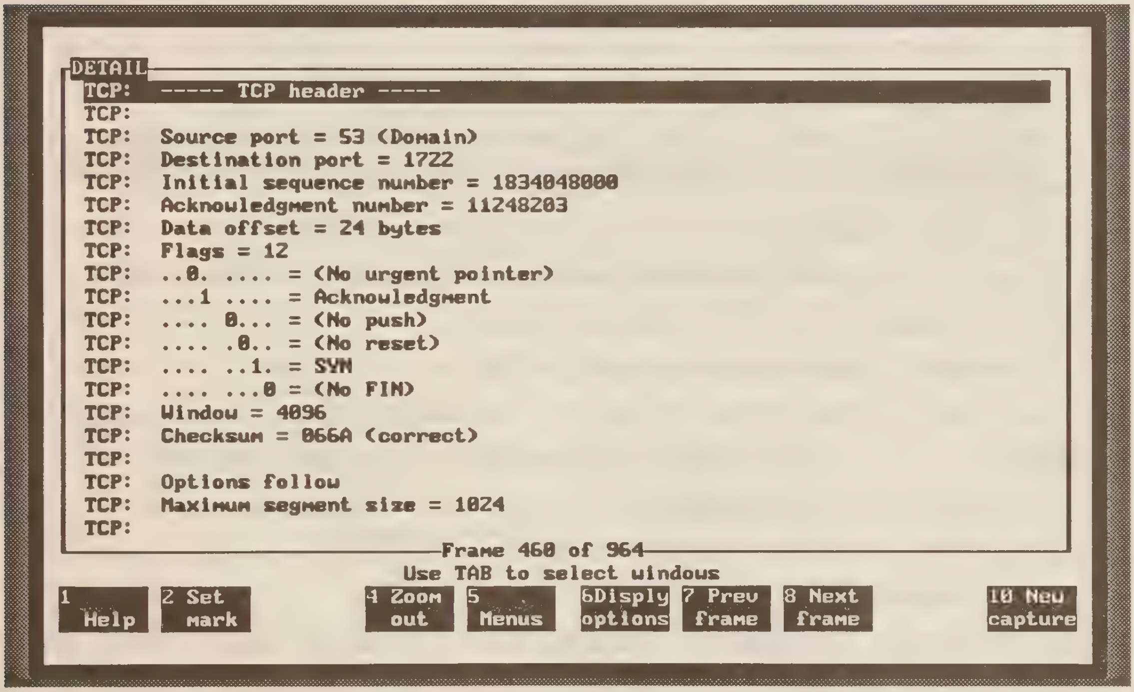
TCPが送るパケットは2つの種類のうちのいずれかである．普通，パケットは緊急ポインタを送らない．これらは通常のデータパケットで，データをもっていることももっていないことも(ACKのみ)ある．もう1種類のパケットは，緊急ポインタをもっているものである．

どんなオペレーティングシステムでも，送るべき緊急データがあるをいうことをカーネルに通



Courtesy of Network General

3-11 TCPヘッダ



Courtesy of Network General

3-12 プッシュ

知することができる手段を備えている．これは多くの場合，システムコールやストリームメッセージである．TCP モジュールがこの緊急データを得ると，それを普通のデータストリームとは別のパケットに入れて送り出す．

そのデータは相手側に着くと，そのコネクション上で待機中のデータ列を飛び越して，そのデータが緊急であるという信号とともにユーザに提示される．緊急データのもつ意味はまったくユーザ次第である．例えば Telnet セッションでは，典型的な緊急データは“現在のコマンドをアボートせよ”である．

緊急データはウィンドウが 0 に設定されているときでさえ送ることができる．送信側がウィンドウを 0 に設定しているときには，受け取ったデータを承認することも可能である．

3.6.1 確認応答戦略

IP ネットワークにおけるフロー制御は TCP モジュールの適切な振舞いに大きく依存している．TCP モジュールは，ノードがネットワークにデータを送りすぎるのを防ぐのに，確認応答機構に依存している．ネットワークに送られるデータが多すぎると，その影響はリモートホストやルータに及ぶ．経路上でルータとして動作している機械や宛先ホストの動作が遅いと，パケットを取りこぼし始める．

取りこぼされたパケットは再送され，ホスト上の負荷の問題をさらにひどくする．さらに悪いことには，こういう状況が発生するとネットワーク自身が混雑し始め，さらに重大な問題に発展してしまう．同様に，一時的なネットワークの不調もまたパケットの消失を招き，多くの再送を引き起こす．

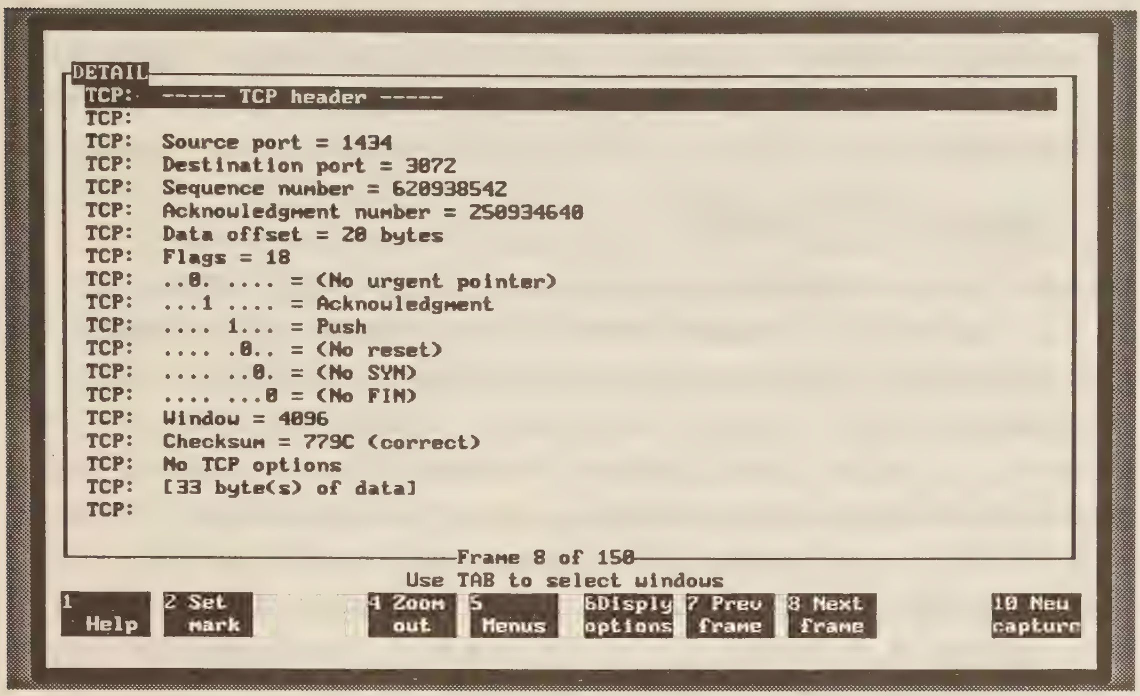
TCP が，保証されたエンド・ツー・エンド転送サービスを提供するためには，確認応答と再送の戦略が非常に重要である．データの再送が積極的すぎても，ネットワークにはよくない．しかし，あまりに簡単に再送をあきらめることは，多くのコネクションを切断してしまうことになる．

TCP においてこれらの問題を扱う基本的な機構は，確認応答とウィンドウの組合せである．確認応答は，まだ受信していないデータの最大番号を示す．

ウィンドウ機構は，相手の送信側が伝送を許される最大のバイト数を表す．ノードがデータを伝送するのを(ほとんど)妨げるためには，単にウィンドウの値を 0 にする．TCP には緊急を通知する機構があるので，こうすれば非常に少量のデータだけが受け取られることになる．

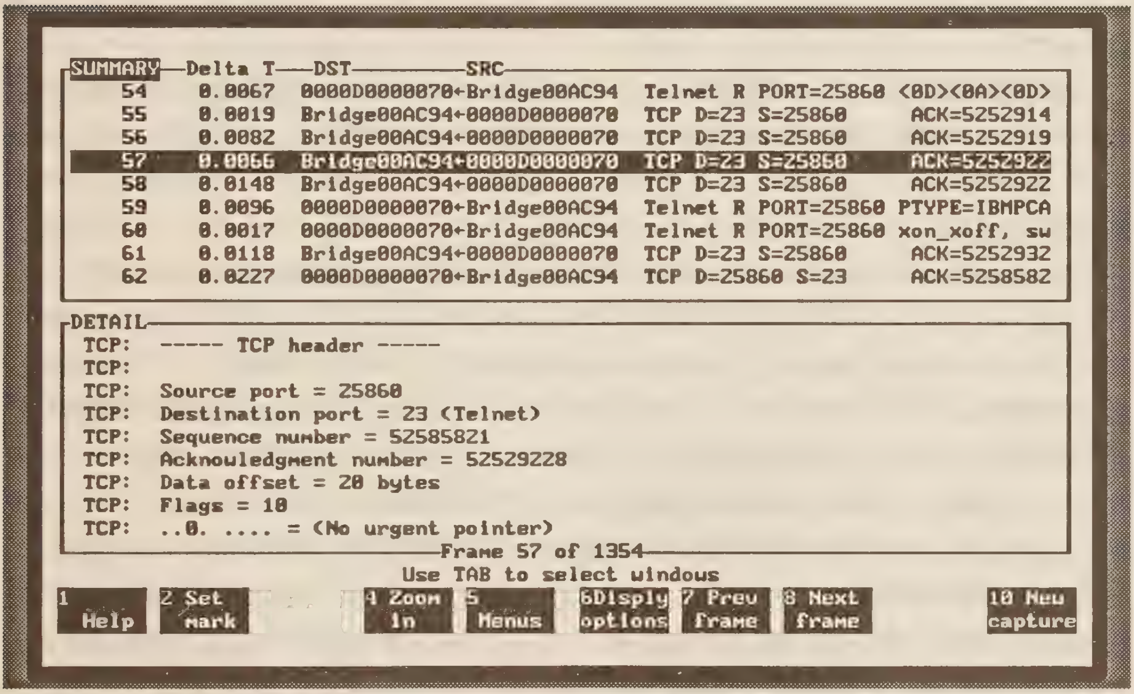
図 3-13 と 3-14 はウィンドウの概念を示している．2 つのノードが確認を交換している．これらのノードを“ブリッジ”と“ゼロ”と呼ぶことにしよう．図 3-13 では，ゼロは 4 つのパケットを送り，その 3 つ目が詳しく示されている．

最初の 3 つのパケットでは確認パケットが次第に増えており，4 つ目で繰返しが休止に達したことがわかる．図 3-14 はブリッジからの応答である．このノードは受け取った 3 つ目のデータパケットを確認している．4 つ目は到着したが処理されていない．これは次のパケットで確認され



Courtesy of Network General

3-13 TCPにおける確認応答



Courtesy of Network General

3-14 TCPにおける確認応答

る．

そのうち安定した状態に達して，コネクションは安定化する．転送の途中で，それぞれのノードが，確認されたデータについて異なる考えをもっているときに，データ転送を適切に処理するための戦略が必要となる．

3.6.2　馬鹿げたウィンドウ症候群

馬鹿げたウィンドウ症候群(silly window syndrome)は，1982 年の RFC で MIT の David Clark 博士によって名づけられた．この状況は TCP モジュールが合理的なウィンドウ管理戦略に見えるものを追いかけるが，意図したものとは反対の結果を得るときに起こる．

ノードは与えられたウィンドウをもっており，それはノードが喜んで受け入れようとするデータの量である．この値は多くの場合，受信側ノードで現在空いているバッファ空間の量である．複数のユーザの間で空間を平均する方法もあるが，さしあたりデータの送信者は 1 つであると想定しよう．そうするとバッファ空間の量は実際に送信側が利用できる空間の量となる．

送信側が提示されたウィンドウを受け取ると，送信側はこの値を用いて利用可能ウィンドウという別の値を計算する．利用可能ウィンドウは，提示されたウィンドウから現在未確認であるデータの量を引いたものである．

受信側が 1000 バイトのウィンドウを提示し，最大伝送長が 200 バイトであるとしよう．送信側はそれぞれが 200 バイトのセグメントを 5 つ送る．これらのパケットは利用可能ウィンドウを使い尽くす．

受信側が最初の 200 バイトを受け取り，データを処理し，そのデータの確認を決定し，実際にまだ利用可能な空間量である 1000 バイトのウィンドウを送り返したとしよう．何といっても，できるだけ大きなウィンドウを提示してスループットを向上させようと努力することは正しいように見える．

提示されたウィンドウ 1000 バイトは送信側で受け取られるが，送信側は 800 バイトのデータがまだ通過中であることを知っている．5 つのパケットが最初に送られて，そのうち最初の 1 つが受信側に届いたのである．800 バイトが通過中で，提示されたウィンドウは 1000 バイトであるので，送信側の計算によると利用可能ウィンドウは 200 バイトのみである．ここまでは特に問題ない．

ここで，さらに 50 バイトが送信側のバッファに到来したとしよう．送信側はこの 50 バイトを送り出す．受信側はこの 50 バイトを受け取り，1000 バイトの提示ウィンドウを送り返す．すると提示されたウィンドウが 1000 バイトで 850 バイトが通過中であるので，利用可能ウィンドウは 150 バイトとなる．次に 1000 バイトのデータを得ると，最初の 150 バイトを送ることになる．ここで 200 という MTU を使うべきであるのに使っていないことに注意してほしい．受信側は 1000 バイトが利用可能だといっているのである．

何が起こったのかというと，データが伝送リンクの MTU に上手にはまらないということであ

る．ときどき，MTU より小さいセグメントを送らなければならなくなる．小さいセグメントを確認することは，人為的にウィンドウを小さくすることになる．ひとたびウィンドウが分割されると，それはデータが流れている間は小さいままである．

送信側が送信を止めると，物事が通常に戻る時間ができる．すべてのデータが確認されて，提示されたウィンドウと同じ大きさの利用可能ウィンドウの最大の状態に戻る．

しかし，大きなファイルを転送している間は，馬鹿げたウィンドウ症候群は容易に劣質化して，ネットワークは小さな断片でいっぱいとなり，それぞれが個々に確認される状態になり得る．当然，ネットワークは詰まって，パケットの紛失と膨大な再伝送が生じることになる．

この問題は送信側と受信側の両方のノードで解決できる．受信側では，ノードが小さなデータパケットを受け取ると，提示するウィンドウをわざと小さくする．こうすると，送信側での利用可能ウィンドウが0になり，送信側は待機することになる．

受信側は十分なデータを受け取ると，人工的な制限を取り除いて，送信側が多くの小さなものではなく大きなウィンドウのジャンプを計算できるようにする．当然，ウィンドウを閉じておく時間が問題となる．もし常に閉じられていれば，スループットはなくなる．常に開いていると，馬鹿げたウィンドウ劣化が起こってしまう．

送信側が助けることもできる．送信側は，提示されたウィンドウに対する利用可能ウィンドウの比率をながめて，この比率が小さい場合，例えば25%のとき送信側はしばらくの間，単に送信を差し控えればよい．

ここでの基本的な問題は，それぞれのセグメントが個々に承認されているということである．即座に確認することは，多くのデータがパイプラインに入ることができるので，よいことである．目標はこうした要求と，多くの小さなパケットがパイプに入ることを妨げる必要とのバランスをとることである．

受信側にとって好ましい一般的規則は，利用可能ウィンドウを増やす目的のとき以外は確認を送るのを抑制することである．この一般的なガイドラインには2つの例外がある．第一に，長く待ちすぎると再送が生じてしまう．そこで待機はタイマ機構に管理されている．第二に，データは逆方向に行くこともあり，そのパケットに確認を相乗りすることは意味のあることである．

確認をどれくらいの間待つか．TCP データストリーム中のプッシュビットがセットされていれば，これは論理レコードの終わりであり，そのあとにおそらく休止が続く．このとき確認は意味をなす．プッシュビットがなければ，50 から 500ms 待機するのが適当であろう．

受信側は確認を制御することができるが，再送間隔は送信側の管理下である．理論的には，送信側は TCP の遅延アルゴリズムを使用しており，それは，再送タイマの計算の基礎として，往復遅延(round-trip delay)の平滑計算を用いている．これによって，再送はネットワークの実際の遅延に基づいて調整され，混雑時にノード速度を低下させることを可能にしている．

3.6.3 不安定時のルーティング

ネットワーク内のあらゆる組合せの間での最適な経路を見つけるためのルーティングアルゴリズムを，ゲートウェイが共同で実現している．あるネットワークやゲートウェイが機能しなくなると，ルーティングデータベースの一時的な不安定が生じる．

ゲートウェイが経路を見つけて安定化しようと試みている間にTCPの実装でタイムアウトが起こると，何が起こるだろうか．ゲートウェイは，良い経路や悪い経路に関する情報を，再方向付けと宛先到達不可能という2つのICMPメッセージを通じてホストにフィードバックする．安定した期間中にこれらのメッセージが届くと，それらは信用できる．しかし，不安定なときには，それらは信用できないこともある．それゆえ，ホストは不安定期間中には，孤立したエラーメッセージに対してある種の懐疑をもって対処しなければならない．

次に問題となる時期は，使用しているゲートウェイが故障したときである．このとき，上記のメッセージは到着することはなく，すべてのデータグラムが失われる．この状況はホストが検出しなければならない．ゲートウェイが何ホップか離れているときには，検出をするのは機能しなくなったゲートウェイの隣のゲートウェイの責任であり，送信ホストには見えない処理である．この問題は，最初のゲートウェイが故障したときにだけ起こる．

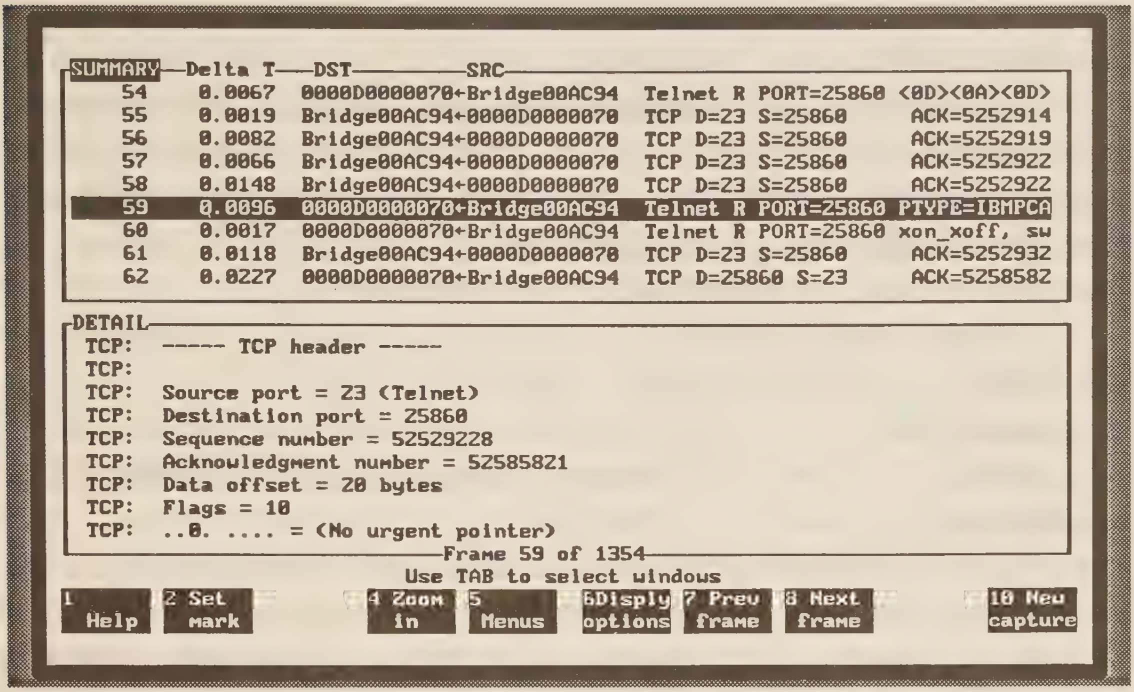
最初のゲートウェイの不在が検出されるものと想定しよう．ホストはゲートウェイをランダムに選び出し，それに向けて送信を開始する．そのランダムなゲートウェイが正しくなければ，そのゲートウェイは再方向づけメッセージを送り返すだろう．もちろん，ランダムに選ばれたゲートウェイもダウンしている可能性もある．ホストはこれを検出して，メッセージが通過するようになるか，試みるゲートウェイがなくなるまでゲートウェイのリストをたどる必要がある．

3.7 コネクションとオプション

新しいTCPコネクションを確立することと，古いコネクションの再同期をとることは同じである．すなわち，全部で3つのパケットが送出される．最初のノードはSYNビットをセットして，使用する初期順序番号を伝達する．すると，他方のノードがこの初期順序番号を承認して自分の初期順序番号を1つ送り返す(図3-15参照)．最後に，最初のノードが相手のISNを確認し，コネクションのデータ転送の準備ができた．

コネクションは単に再同期されるだけなので，ユーザデータを最初のパケットに入れて運び，トランスポート層がコネクションを確立するのと同時にアプリケーションを初期化することも可能である．また，オプションの調整を行うのに，これら最初のいくつかのパケットを使用することも可能である．

図3-15では，最大セグメント長を1024に設定するためのオプションがパケットに含まれてい



Courtesy of Network General

3-15　TCPオプションのネゴシエーション

るのが示されている．この戦略により，それぞれの TCP パケットは IP パケットの中に収まり，そしてこの IP パケットは 1 つのイーサネットパケットの中に収まることになる．

MTU の設定は単に，トランスポート層が IP 層に渡すデータ長を設定することである．IP はデータの断片化について独自の選択を行うことができる．IP 実装が処理できなければならない最小のパケット長は 576 バイトである．トランスポート MTU が 1024 バイトであっても，そのパケットは遅いシリアル回線を通過しなければならないと IP が判断すると，そのパケットを 2 つに断片化することもできる．

3.7.1　TCP の拡張

非常に広帯域のネットワークが出現したことにより，TCP は古くなり始めている．300bps から 800Mbps までの範囲では，TCP はきわめて良好に動作する．非常に広帯域で大きな往復遅延があるリングにおいて問題が生じ始める．典型的な例はもちろん，Internet の核を拡張するために導入された広域ギガビットリンクであろう．

ビット／秒で表されるリンクの帯域幅と，秒で表される往復遅延とから，パイプを満たすのに必要なバイト数を得ることができる．パイプをいっぱいにするには，少なくともその程度の量のデータが未確認となることが許されるような十分な大きさのウィンドウが必要となる．

パイプが大きな数(10^5 ビット以上)になり始めると，TCP は問題を抱え始める．この状況を表す

名前は Van Jacobson と Robert Braden により，“長太パイプネットワーク”(long, fat pipe network : LFN) と提案された.

長太パイプの簡単な例として，T1 速度(1.5Mbps)衛星チャネルがある．この LFN では帯域幅と遅延の積が簡単に 10^6 ビット以上になってしまう．これは，1200 バイトの未確認 TCP セグメント 100 個分に相当する．地上の経路でも，この種の問題を生じることがある．DS3(45Mbps)上の 30ms の国内横断遅延も 10^6 ビットのパイプ長を越える．

基本的な TCP プロトコルには実際問題として 3 つの制限がある．

・ウィンドウサイズの制限

・一括確認

・往復遅延の計測

これらの制限は 3 つとも，コネクションを確立するときに交渉されるオプションの問題である(図 3-16 参照)．

ウィンドウサイズの制限は最も大きなものである．確認され得るバイト数を表すために 16 ビットのフィールドが用意されている．このため可能な最大ウィンドウは 64 キロバイトである．しかし，現在の TCP 実装では 16 ビット全部が使われると動作しないものが多いので，より現実的な制限は 32 キロバイトである．パイプが 10^6 ビットの場合，ウィンドウが 32 キロバイトだと，パイプはその容量のたった 1/4 だけしか満たされないことになる．

TCP オプションは，コネクション確立時に“暗黙スケールファクタ(implicit scale factor)”を許すことにより，この問題を解決している．このオプションにより，ユーザは提示されたウィン

名前	#	長さ	説明
オプションリストの終わり	0	―	
動作なし	1	―	動作なし．
最大セグメント長	2	16ビット	初期コネクション要求でのみ送られなければならない．
ウィンドウスケールオプション	3	3	データは，受信側が真のウィンドウ受信値を右シフトするときのビット数を表すシフトカウントパラメータである．
SACK許可	4	2	選択ACK許可．初期SYNで送られなければならない．
SACKオプション	5	可変長	受信された隣接した順序番号のブロックのリストを含む．それぞれのブロックは2つの16ビット整数に基づく．それは相対的起源(ウィンドウの左側にあるTCPヘッダ中の最後のACK番号に関連する)とブロックサイズである．TCPオプション用に44バイトが利用できる．それゆえ，10ブロックの選択的確認が許される．他のオプションを使うと，SACKに利用できる空間は小さくなる．スケーリングが効果的であれば，SACKフィールドはウィンドウと同じファクタによってスケールされる．
TCPエコーオプション	6	4バイト	エコーされるデータ．往復遅延計測のために使われるフィールドはタイムスタンプを含む．
エコー応答	7	4バイト	

3-16　TCPオプションコード

ドウにスケールファクタを乗じて真のウィンドウサイズとすることができる.

2番目の問題は一括確認である. デフォルトのTCP確認では, ある1つのデータに問題が生じた場合, 他のパケットに問題がなくても, 送信側は確認されていない点までのすべてのパケットを再送しなければならない.

選択的確認の問題を扱うために, 1つのTCPオプションが追加された. 選択的確認(SACK)は2つの段階を経て用いられる. 第1に, コネクションを確立するときに, 両側でSACKを使用することについて同意しなければならない. 第2に, 将来のTCPパケットにおけるオプションとして, 受信側は確認しようとするデータの長さとオフセットを記述したSACKを含める.

デフォルトのTCPでの3番目の基本的な問題は, 往復遅延計測(round-trip timing : RTT)である. ほとんどのTCP実行はRTTを再送タイマの基準値に用いている. ネットワークが本当に混雑したとき適切に対処するためには正確なRTTが必要とされる.

ほとんどの実装では, RTTはパケットが出発した時間とパケットが確認された時間の差として測定される. TCPには一括承認があるので, ほとんどのRTTタイマはウィンドウあたりの1回の計測に基づいている. これは小さなウィンドウではかまわないが, ウィンドウが非常に大きくなると問題が生じ得る. RTT測定の解決法はエコーオプションである. エコーオプション中のデータは即座に送り返され, 往復遅延計測を無視できるコストで得ることができる.

基本的なTCPの仕様では, 任意の時点で送信されるオプションに関するよく定義された概念はない. 実際, オプションは本当は1つだけしかなく, それはリンクが同期されるときにMTUを設定するオプションである. 新しいオプションを定義するとき, もちろん現在の実装が動作するのを保つことが1つの目標となる.

これを達成するためには, あらゆるTCPオプションは最初に取り決められなければならない. ひとたびコネクションが確立されると追加することはできない. これは, 貧弱な実装はおそらく最初のパケットに余分な追加オプションがあっても無視するが, それ以降のパケットでそれらを見つけると動作しなくなるだろうという考え方に基づくものである. 最初にひとたび同意がなされると, 通信している2つのTCPモジュールはその拡張を用いることができる.

3.7.2 発信元抑制(quench)とTCP

混雑に対抗するための武器として認められているのは, ICMP発信元抑制メッセージである. ゲートウェイやホストに送られてくるデータが速すぎると, もとのデータを送ったホストにこのメッセージを送る. ホストがこのメッセージを受け取ると, 発信元抑制を送ったシステム宛のトラフィックの量を減らそうとするだろう.

混雑には2つの主な原因がある.

- タイニグラム(tinygrams)
- 発信元抑制のためのサポート欠如

タイニグラムはデータをほとんど含まず，多くの場合 Telnet のような 1 文字ごと (character-at-a-time) のアプリケーションによって生成される．TCP は前のデータが確認されるまで新しいデータを送るのを止めることにとって，タイニグラムを減らすことができる．

もう 1 つの問題は発信元抑制に対して適応しないことである．通常の動作では，TCP モジュールは混雑があるのを見て，提示された TCP ウィンドウのうちで満たす量を減らす．相手の TCP が現在 4096 オクテットのウィンドウを提示し，こちらの TCP が発信元抑制を受け取ると，ウィンドウサイズがあたかも実際には 1024 バイトであるかのように振る舞い始める (相手側は 4096 を示し続けるが)．しばらくして (例えば受信した ACK の数を数える)，ローカルな TCP は再び真のウィンドウサイズを使用し始める．

他の 2 つの方式は，エンド・ツー・エンドの層でタイムアウトを用いる Raj Jain の混雑制御 (CUTE) と，Van Jacobson の低速開始法である．これらの方法では，混雑が軽減されるまでウィンドウを小さくする代わりに，混雑が発生するまでウィンドウをゆっくりと大きくしていく．混雑が発生するとウィンドウを最小値にし，混雑が再び発生するまで再びゆっくりと大きくしていく．これらの方法では，両方とも，発信元抑制の方法を用いるのではなく，紛失したパケット数を用いて混雑が存在しているのを検出する．ここでの論理は，安定した状態を達成するためにゆっくりと負荷を築いていくのは，回路をたたいて発信元抑制を強制するよりもよいというものである．後者のアプローチはインターネットワークの混雑崩壊を引き起こしやすい．

3.8 TCP と OSI

トランスポートサービスを提供するもう 1 つの ISO のトランスポートプロトコルがある．これは一連の 5 つの異なるトランスポートプロトコルであり，それぞれが別のレベルのサービスを提供している．TP クラス 4 (TP4 として知られる) は最高上位レベルのサービスを OSI アプリケーションに提供し，それは機能的には TCP プロトコルに等しい (図 3-17 参照)．

OSI の世界では，TP4 トランスポートモジュールはコネクション指向ネットワークサービス (CONS) とコネクションレスネットワークサービス (CLNS) という 2 つのネットワークサービスのうちの 1 つを用いる．CLNS は実質的に IP に等しく，TP4 は実質的に TCP に等しい．

Sun ワークステーションは 2 つの方法のいずれかにより，これらのサービスにアクセスすることができる．TCP/IP を用いない OSI プロトコルそのものの実装をアプリケーション込みで購入することができる．あるいは，SunLink ゲートウェイをネットワーク上に置くこともできる．この SunLink ゲートウェイは他のサーバとちょうど同じようなサーバである．ワークステーションはこのサーバに接続して FTAM のような基本的な OSI アプリケーションにアクセスできる．この意味でこのゲートウェイは 2 つの異なるレベルでルーティングを行っているといえる．それは 2 つの環境のトランスポートおよびネットワーク層を互いに接続するとともに，2 つのアプリケー

種類	TCP	TP4
データ転送	ストリーム	ブロック
フロー制御	バイト	セグメント
誤り検出	チェックサム	チェックサム
アドレッシング	16ビットのポート番号	可変TSAP-ID
割込み	緊急ポインタ	急送データ
セキュリティ	11バイトのIPフィールド	可変TP，IPフィールド
優先度	3ビットのIPフィールド	16ビットのTP，IPフィールド
データグラム	UDP	ISO 8602
切断	礼儀正しい	中断

出典: Transport Protocols for Department of Defense Data Networks: Report to the Department of Defense and the National Bureau of Standards, National Academy Press, February 1985. 表の写し元は David M. PiscitelloとA.Lyman Chapin, ``TCP and TP4: Moving Forward,," pp.2-9, ConneXions, Vo.4, No.9, September 1990.

3-17 TCPとTP4の比較

ションも互いに接続する．

3.8.1 TCP 上の TP 0

OSI アプリケーションを動作させる必要があるが OSI ネットワークをもっていないときはどうするか．何人かの Internet アプリケーション設計者は OSI の上位層，特にセション層とプレゼンテーション層で提供されるサービスのいくつかを欲しいと思ったが，OSI ネットワークを待つことはできなかった．

TCP と TP4 はともに同じ基本サービス――保証された誤りのないデータ伝送――を提供するので，スタックをつなぎ合わせられないという理由はない．TP0 トランスポートプロトコルは下位にコネクション指向ネットワークサービス(例えば X.25)を想定している．TCP の上に TP0 を継ぎ合わせることは，単なるコネクション指向ネットワークサービスではなく，信頼性のあるものを創造し，OSI の TP4 と同じ意味構造を生み出す．

この手続きは RFC1006 で定義されている．TCP ポート 102 はそのユーザである TP0 のために予約されている．これらのプロトコルの間にはいくつかの小さな相違があり，解決されなければならない．例えば TCP にはポートの概念があるが，TP にはない．ポートの指示が来るのを待つのではなく，TCP/IP に基礎を置く相手側の OSI アプリケーションは listen を発行しなければならない．

RFC1006 の主なユーザは，Performance Systems International の Marshall Rose が開発した ISO 開発環境(ISODE)である(図 3-18 参照)．ISODE はセション層からアプリケーション層までの主要な OSI 層を実行する．それはいくつかのアプリケーションを内蔵しており，そのうちの 1 つ X.500 ディレクトリの実装は最も有名である．他のアプリケーション開発者達もプログラム開発

```
DETAIL
 TCP:   ----- TCP header -----
 TCP:
 TCP:   Source port = 1082
 TCP:   Destination port = 102 (ISO)
 TCP:   Sequence number = 1065688
 TCP:   Acknowledgment number = 38800
 TCP:   Data offset = 20 bytes
 TCP:   Flags = 18
 TCP:   ..0. .... = (No urgent pointer)
 TCP:   ...1 .... = Acknowledgment
 TCP:   .... 1... = Push
 TCP:   .... .0.. = (No reset)
 TCP:   .... ..0. = (No SYN)
 TCP:   .... ...0 = (No FIN)
 TCP:   Window = 4096
 TCP:   Checksum = A8BA (correct)
 TCP:   No TCP options
 TCP:   [6 byte(s) of data]
 ISO_DE: ----- ISO Development Environment -----
 ISO_DE:
 ISO_DE: Multi-frame TPDU: frames 10, 11, 12
 ISO_DE: Version = 3
 ISO_DE: Packet length = 132
 ISO_DE:
 ISO_TP: ----- ISO Transport Layer -----
 ISO_TP:
 ISO_TP: Header length = 2
 ISO_TP: TPDU type = F (Data)
 ISO_TP: EOT: Last TPDU of a sequence
 ISO_SS: ----- ISO Session Layer -----
 ISO_SS:
 ISO_SS: SPDU type = 13 (Connect)
 ISO_SS: Length of SPDU parameter field = 123
 ISO_SS: -- Connection identifier parameter group (length = 28)
 ISO_SS: Calling SS-user reference = <1407>kurt380
 ISO_SS: Common reference = <170D>881109193935Z
 ISO_SS: -- Connect/Accept item parameter group (length = 12)
 ISO_SS: Protocol options = 00
 ISO_SS:         .... ...0 = Not able to receive extended concatenated SPDUs
 ISO_SS: Maximum TSDU size = 65528 (Initiator to responder)
 ISO_SS: Maximum TSDU size = 65528 (Responder to initiator)
 ISO_SS: Version number = 1
 ISO_SS: Session user requirements = 0002
 ISO_SS:       .... 0...  .... .... = No symmetric synchronize
 ISO_SS:       .... .0..  .... .... = No typed data
 ISO_SS:       .... ..0.  .... .... = No exceptions
 ISO_SS:       .... ...0  .... .... = No capability data
 ISO_SS:       .... ....  0... .... = No negotiated release
 ISO_SS:       .... ....  .0.. .... = No activity management
                              Frame 10 of 203
                          Use TAB to select windows
1      2 Set        4 Zoom  5       6Disply 7 Prev  8 Next          10 New
 Help    mark         out    Menus  options  frame   frame          capture
```

Courtesy of Network General

3-18 ISODE

にISODEを用いている(ここでもX.500ディレクトリは最も有名である).

ISODEから学ぶ教訓は，ネットワークアーキテクチャは不可侵ではないということである．プロトコルは必要に応じて様々なアーキテクチャから引き抜かれる．これが重要な教訓であるのは，多くの人々がOSIへの“移行”を考えてきたのに対してISODEは共存が実際に可能であることを示したからである．

3.9 UDP

ユーザデータグラムプロトコル(UDP)はTCPの簡単な代替品である．それはIPの上に構成され，ごく少ない価値しか加えていない．実質的には，UDPはIPの単なるユーザインターフェイスである．特に，配達の保証やパケットの重複に対する防護はUDPの機能ではない．

では，どうしてアプリケーションがUDPを利用しようとするのだろうか．それは，UDPはTCPに比べて，いくつかの利点をもっているからである．1つ目は，UDPはコネクション指向(connection-oriented)ではないので，下位にあるIPのマルチキャスト機能を用いて1つのUDPパケットを多数のホストに送ることができる．2つ目は，UDPはTCPほど資源を使わない．コネクションを設定したり伝達の保証を行う必要がないので，用いる資源が少ないからである．イーサネットのような，パケット廃棄率と重複率の低い環境ではUDPが適切かもしれない．3つ目は，UDPの実装はより簡単に行えるので，使うコードが少なくて済む．これは，SNMPのような限られた資源しかもっていない装置で動作するプロトコルには重要な要素である．4つ目に，UDPはNFSのような多数のクライアントをサポートする機能を実装するのに用いることができる．

いずれの場合でも，UDPを利用するアプリケーションはデータの配達を心配しなければならない．一部のアプリケーションは心配しない：もしパケットが到着しなければ，何も起こらない．要求(request)に対して応答できないとき，パケットを再伝送するリクエスト-応答の手法(scenario)を用いるアプリケーションもある．ある要求が送られたら，そのプロトコルはある時間内に応答が受けられると指示する．再送タイマが切れたら，その要求は再送される．これは，NFSがUDPの上に構成されたとき使われる方法である．

UDPのパケット方式は非常に簡単である．TCPのように，各パケットはそれぞれ2バイトの送信ポートと着信ポートのアドレスをもつ．送信ポートは省略可能であり，0にすることもできる．アドレスの他に，UDPパケットには長さの記述とチェックサム(check-sum)があり，その後にはデータがある．長さにはヘッダ長とデータ長が含まれているので，正当なパケット長の最小値は8である．最大値は8キロバイトである．チェックサムは16ビットの1の補数の和の1の補数である．このとき0の値は許され，それは送信側がチェックサムを発生しなかったことを意味する．

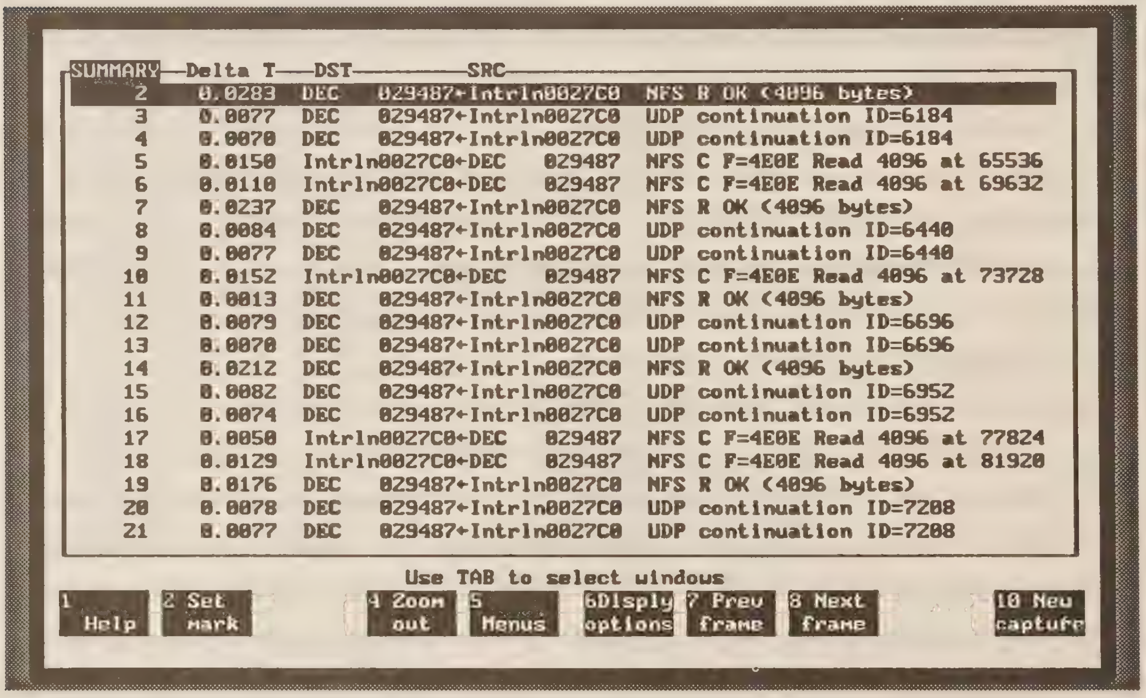
3.10 RIP

ここで伝送プロトコルから経路制御に話題を移そう．経路制御プロトコルには2つの種類がある．1つのルーティング領域では内部プロトコル(interior protocol)が使われ，ルーティング領域どうしの間には外部(exterior)プロトコルが使われる．まず，内部プロトコルから始め，その後でEGPのような外部プロトコルについて述べる．

ルーティング情報プロトコル(Routing Information Protocol : RIP)は内部ルーティングプロトコルの例であり，多くのSunのネットワークを含めて，バークレー4.2 Unixの系統に属するネットワークで広く用いられている．RIPはUDP伝送プロトコルの上に構成されている．

RIPには多少の問題があるものの，簡単なトポロジーをもつ小さなネットワークでは役に立つ．RIPは距離ベクトルアルゴリズム(distance vector algorithm)の例である．距離はホップの概念に基づいている．典型的には，1つのホップは1つのサブネットワークである．遅いリンクにより大きなホップ数を与えることもできる．RIPでは1つのノードへの総ホップ数は15に制限されている．つまり，16は無限と見なされ，到達不能と判断される．

これを距離ベクトルアルゴリズムと呼ぶのは，用いるすべてのリンクを知るわけではないからである．知っているのは使われる隣のノード（ベクトルの始点）と総距離だけである．



Courtesy of Network General

3-19 UDP継続

各ルータは周期的に隣接するルータにRIPパケットを送る．そのパケットには，そのルータが到達できるすべてのネットワークとホストが記述されている．ある特定のルータは他のいくつものルータからRIPパケットをもらう．これらのルータからある特定のホストまでの距離を比較すれば，最適な経路を決定することができる．

ここで，各ルータがそれぞれの知っている最適な経路を知らせてくれることに依存していることに注意せよ．同様に，他のルータに自分の知っている最適な経路を教えなければならない．これは，自分の隣人に自分の知っている世界を教えることに似ている．安定した状況ではこれはうまく動作する．しかし，不安定なトポロジーにおいては，各ノードが最適な経路を判断するには少し時間がかかる可能性がある．

これらの制限にもかかわらず，距離ベクトルアルゴリズムはまだ広く使われている．その1つの理由はこのアルゴリズムがすでに広く実装されているということである．距離ベクトルアルゴリズムは1969年に既にARPANETで使われていた．しかし，RIPの祖先はXNS RIPからである．RIPはXNS RIPに非常に似ていて，XNSアドレスの代わりに一般的なアドレスが使われ，ルーティングの更新が30秒に1回に制限されているだけである．

RIPは固定された指標(metric)に基づいている．その指標はホップであり，そして各リンクのコストは固定されている．これは遅延，信頼性や負荷などの実測値のようなリアルタイムパラメータに基づいて経路を選択することはできないことを意味する．

15という最大距離の制限は，コストを割り当てるのに本格的な制限を課している．例えば100Mbpsで動作するFDDIネットワークと9.6kbpsで動作する電話回線があるとしよう．電話回線はFDDIリンクより10 000倍も遅いので，10 000倍のコストを与えるのが合理的である．15のコスト空間で実現するのは困難である．

RIPはルータによって管理される5つの情報に基づいている．

・宛先IPアドレス
・ゲートウェイIPアドレス
・ゲートウェイに到達するための物理ネットワークインターフェイス
・距離指標の現在値
・このエントリが更新されてからの経過時間

この情報は，基本的には転送先に関するデータベースである．自分と異なるサブネットワークのネットワークアドレスが与えられたら，ルータはデータベースを参照して，どのゲートウェイを使うかを見い出す．もしルータがより良い指標をもつRIPパケットを受け取った場合には，データベースにある現有のエントリを新しいエントリに置き換える．

もしその経路がより悪くなったらどうすればいいか．もしそのパケットが利用しようとするゲートウェイはコストが悪くなったというなら，それを信じて指標を更新すべきである．他のノードがより悪い経路をもっていても，それは無視する．

ルータがクラッシュした場合にはどのようにすればいいか．RIPの世界では，隣接ルータは30秒ごとに更新されている．もし隣接のルータが特定の経路の更新情報を180秒間送ってこないなら，その経路は無効とされる．ある特定のネットワークが（少なくともある特定のルータの視野で）到達不可能と判断されたら，その指標は16（無限）に設定される．

この方法で起こり得る1つの問題は，2つのノードが互いに相手の経路が最適であると判断するとき発生する．これを予防するために，水平線分割(split horizon)という手法が使われている．この手法では，更新情報はその更新情報を送ってこなかった隣接ノードだけに送出する．

水平線分割の1つのバージョンとして，有毒反転法(poisoned reverse)では，更新情報にすべての経路を含めるが，その指標を無限にする．AとBという2つのノードを考えよう．両方とも同じサブネットワークの上にあって，ノードDに到達できる．AがBに更新情報を送るとき，Dが存在して，そのコストは無限であると伝える．つまり，Bは直接Dに行けるので，Aを経由するべきではない．この有毒反転法はルーティングループを解除する1つの簡単な方法である．2つのルートが互いに相手を指している場合に，コストを16に設定することによってそのループをすぐに解除することができる．これに対してその経路が放送されない場合には，その経路を除去するのにタイムアウトを待たなければならない．

有毒反転を用いた水平線分割では2つのゲートウェイの間の経路ループの発生を防止できるが，3つのゲートウェイが同様な状態に落ち込む場合がある．このとき，それを解決するための唯一の方法はこの3つのゲートウェイすべてに無限経路の情報を送り出すことである．これは，あるネットワークが到達不可能となったときに更新を起動することによって行われている．

RIPはポート520を用いたUDP伝送プロトコルに基づいている．すべてのルーティング更新情報はこのポートから発生し，そしてすべての自発的ルーティング更新情報はこのポートに行く．要求に対する応答はその発信源のポートに送られる．

いくつかのノードを無言(silent)RIPとして実装し，ネットワークを聴くだけにすることが可能である．ホストはこのようにするかもしれないし，初期化を行ったばかりのゲートウェイが，隣接のノードに非信頼的な情報を送らないように，ある時間内ネットワークを聴くこともあろう．

RIPはいろいろなネットワーク層のプロトコルのルーティング情報を伝送することが可能であるが，現在はInternetプロトコルのために定義されたアドレス族しかもっていない．RIPが帯域幅の使用者であり，大型のネットワークに対応できないことから，他のアドレスの族を含むように拡張されることはないであろう．

RIPのパケットは最大512バイトのデータグラムサイズに制限されており，すべてのInternetのホストが扱える576バイトのパケットサイズに適合できる．これによって，ルーティングパケットが分割されないことが保証できる．長いルーティングテーブルは，いくつかのパケットで伝送することができる．

RIPパケットのエントリはアドレスによって識別されている．そのアドレスは，あるホストや

ネットワークあるいはデフォルトアドレスを意味する特別符号を参照することができる．もしあるゲートウェイを通してすべてのホストがアクセスできるなら，これらのホストを個別に記述する必要はない．実際に，一部の実装ではホストの経路をサポートしていないので，それを受け取ったら廃棄することになる．

ネットワーク番号はサブネットワーク番号でもよい．基本的な仮定は，最も具体的なエントリから始め，一般的なエントリに移っていくことである．最初はホストエントリを探し，次はサブネットのエントリ，次はネットワーク番号，そして最後はデフォルト経路の順で行く．

ネットワーク番号の解釈は，自ホストのサブネットワークマスクをかけることによって行う．いま，255.255.255.0のサブネットマスクをもっているとしよう．255は2進数では11111111であり，マスクのその部分のすべてのビットが“オン”にされたことを意味している．このマスクに基づくと，128.6.0.0はネットワーク番号であり，128.6.4.0はサブネットワーク番号であり，そして128.6.4.1はホストアドレスである．ここで，そのネットワークのサブネットワークマスクをもっていないと，サブネットワーク番号は意味をもたないことに注意せよ．このため，ホストは，相手のホストが同じネットワーク内に存在しないと，サブネットワークアドレスを送ることはしない．それを他のネットワークに送るのは愚かなことである．

サブネットワークのフィルタリングは境界にあるゲートウェイによって行われる．内部では，各サブネットは個別のネットワークである．境界にあるゲートウェイはネットワーク全体として1つのエントリしか送らない．これと同じ論理で，1つの境界ゲートウェイは他のネットワークと通信するとき，単一のホストの情報は含めない．すなわち，境界ゲートウェイは1つのネットワークエントリとして広告(advertise)するべきである．

0.0.0.0というアドレスはデフォルト経路である．デフォルト経路はルーティングテーブルのエントリに合わないすべてのパケットを，より“賢い”ゲートウェイに誘導する．われわれはどのようにしてデフォルトとなるゲートウェイを決めるべきであろうか．1つの規則としてはEGPのような，外部ゲートウェイプロトコルを使っているゲートウェイはデフォルトゲートウェイである．もし複数のEGPゲートウェイが存在すると，指標によって，あるゲートウェイが“好ましいデフォルト”(preferred default)となる．典型的には，各々の自律システム(autonomus system : AS)はそれぞれ固有の好ましいデフォルトをもっていて，0.0.0.0の経路はASの境界から出ることはない．

RIPはタイマの概念に基づいている．1つのノードは30秒ごとにすべての隣接ゲートウェイに対して完全なRIP応答を送らなければならない．隣接ゲートウェイのタイマのほかに，RIPのモジュールは各経路に対して2つのタイマを管理している．

- タイムアウト値
- ゴミ回収タイマ

ある経路がタイムアウトになる（タイムアウト値が切れる）と，その経路は無効になる．しかし，

そのエントリはしばらく保持されて，次の更新パケットに含まれる．ゴミ回収タイマがタイムアウトになると，その経路は除去される．

タイムアウトタイマの値は180秒である．タイムアウトが起こる，あるいは指標16のエントリを受けると，ゴミ回収タイマが120秒に設定される．このときもしその指標が16でなければ，16に設定される．ここであるエントリが変わると，それによってフラグが設定されることに注意せよ．このフラグは出力プロセスの応答を起動し，隣接のノードにこの新しい情報を知らせる．

3.10.1 RIP要求

通常，要求は放送(broadcast)として送られる．もし，その要求がポート520からのものであるなら，それはすべてのアクティブなRIPプロセスに対するものであり，無言RIPは静粛を保ち続けるべきである．もしそのリクエストがポート520以外のポートからのものであるなら，無言RIPも応答すべきである．

リクエストはエントリごとに処理される．エントリがない場合には応答は生成されない．アドレス族が0（不特定）で指標が16であるものは特別なエントリであり，ルーティングテーブルを全部送ることを意味している．この場合を除いて，処理は簡単である．RIPモジュールはリクエストを1行ずつ，転送データベースと比較する．転送データベースに経路が存在すれば，その指標をパケットに入れる．そうでなければ16の値を入れる．すべての応答が記入されたら，そのデータグラムを発信ポートに送り返す．

ここで遠隔ノードが特定なエントリを要求した場合には，RIPモジュールはその情報しか答えない．しかし，テーブル全体が処理される場合，サブネット隠蔽と有毒領域分割の通常の法則が適用される．その考え方は，特定なホストやネットワークが要求された場合には，それは（多分人間による）診断的(diagnostic)な要求であるので，要求された内容を知らせるべきである．一方，その遠隔システムがブート中であり，したがってRIP要求を送っているなら，サブネット隠蔽法と有毒法を用いるべきである．

3.10.2 応答処理

応答は3つの原因で受信される．

- 質問に対する応答
- 定期的な情報更新
- 指標の変更により起動された情報更新

応答の処理は常に同じである．もしある応答がポート520からのものではないなら，その応答は無視される．もし応答IPの発信元アドレス(source address)が有効な直結隣接ノードでなければ，それも無視される．放送ネットワークでは，送信ノードが自分の応答を受信することがあるので，自分自身からのものであるか否か調べることも重要である．

初期の正常性調査が完了すると，個々のエントリが調べられる．16より大きな値をもつエントリは無視される．次には，プロトコル族のID(family ID)が適切であるかどうか調べる．次いで，アドレスそのものの適切性についても調べる．もしそのアドレスがネットワーク127(ループバックネットワーク）のものであるなら無視される．放送アドレスとホスト経路をサポートしていないノードの場合，ホスト経路も無視される．

次に，RIPのフィールドにある指標が応答の到着したネットワークのコストを加えることによって更新される．もしその和が16を越えれば，16の値が使われる．この時点で，ルーティングデータベースからRIPエントリ中のアドレスが検索される．もしそのアドレスが存在しない場合，ルーティングデータベースに追加する(多くの場合)．もちろん，無限指標をもつ経路は追加されない．また，データベースに同等もしくはそれよりよいサブネット経路がすでに存在するとき，ホスト指定経路も追加されない．

エントリが追加される場合，次の4つのステップが必要となる．

・アドレスを追加して，距離を設定する
・ゲートウェイをデータグラムの発信元として登録する
・タイムアウトを初期化する
・経路変更フラグを設定して，更新を起動する

経路がすでに存在する場合には，転送データベースに記述されたゲートウェイとRIP応答に記述されたゲートウェイが比較される．もしこの2つのゲートウェイが同じであれば，ルーティングデータベースの情報は更新され，タイマがリセットされる．もしその新しい指標が16であれば，ゴミ回収処理が開始され，更新作業が起動される．もしゲートウェイが異なっており，新しい指標がよりよい値をもつなら，そのゲートウェイがデータベースに追加される．

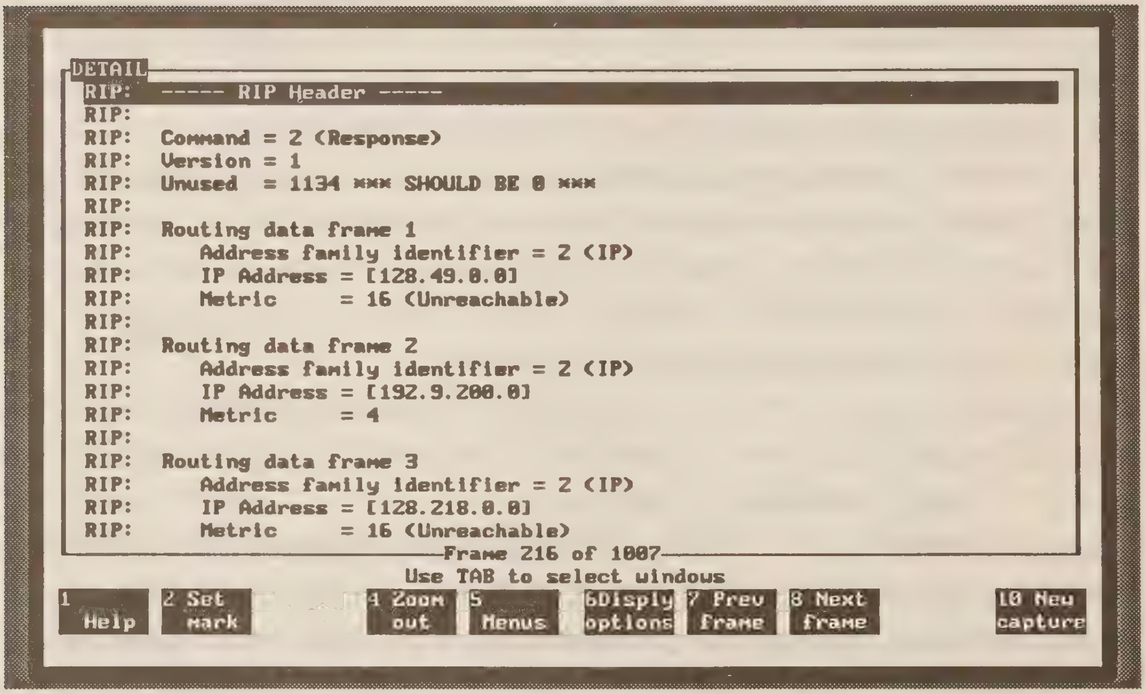
もしゲートウェイが異なっていて，両方の指標が同じ値をもつなら，通常はそれは無視される．しかし4.3BSDにおけるRIPルートデーモンの実装では，その情報が使われる．もし現有の経路がタイムアウトになりそうなら，到着したばかりのゲートウェイ（したがってしばらくは有効である)を使うことは合理的である．1つの考え方としては，もしそのタイムアウトが少なくとも半分過ぎているなら，等価な経路に切り換えるのが合理的である．

すべての新しいエントリがデータベースに合併されると，RIPモジュールは隣接ノードにそれを知らせなければならない．パケットは要求に対する応答，30秒ごと，あるいはあるエントリの更新によって起動される情報更新のとき送出される．

起動された情報更新は，ネットワークに対する衝撃(bombard)を起こさないように特別な取扱いが必要である．ある起動された情報更新が送られたら，タイマが1から5秒間までの間のランダムな値に設定される．この期間内は，他の更新情報は送られない．なお，通常の情報更新が，起動された情報更新タイマが終了する前に予定されているなら，起動された情報更新は常に抑制される．

固定テーブル	
コマンド	1バイト
	1：要求（すべて/一部のルーティングテーブルを送る）
	2：応答（リクエストや問合せ (poll) に対する応答）
	3：トレースオン（現在では用いられていない）
	4：トレースオフ（現在では用いられていない）
	5：予約（Sun Microsystemsによって使われる）
バージョン	1バイト
必ずゼロ	2バイト
可変テーブル	
アドレス族ID	2バイト
必ずゼロ	2バイト
IPアドレス	4バイト
必ずゼロ	4バイト
必ずゼロ	4バイト
指標	4バイト

3-20 RIPのパケットフォーマット



Courtesy of Network General

3-21 1つのRIPパケット

バークレー版の routed デーモンには 2 つの管理制御が付加されている．1 つ目は，あるホストの隣接ノードの一覧表が記載されている隣接ノード表が保管されていることである．もしその隣接ノード表が有効であるなら，RIP モジュールは表に載っているホストからの応答しか受理しない．2 つ目の付加は，特定の宛先を許可したり不許可にしたりするものである．特定なインターフェイスの入力あるいは出力方向ごとに 1 つの表が対応づけられる．RIP モジュールは表に載っている目的地を取り除いたり取り入れたりする．すなわち，それ以外の宛先だけが認められたり認められなかったりする．

図 3-20 に RIP パケットのフォーマットを示す．図中の 2 つのコマンドはリクエストと応答である．可変な部分には目的地のアドレス，ゲートウェイの IP アドレス，目的地に到着するためのコストが格納されている．図 3-21 に 1 つの典型的なパケットを示している．

3.11 OSPF と IS-IS

開放型最短経路優先方式(Open Shortest Path First : OSPF)は，非常に大きなルーティング領域を取り扱うために開発された 2 つのプロトコルのうちの 1 つである．この方式は TCP/IP の環境で使われるための内部ゲートウェイプロトコル(Internal Gateway Protocol : IGP)であるが，RIP のようなプロトコルよりずっと洗練されている．

OSPF はリンク状態ルーティングアルゴリズムに基づいているため，ネットワークを過多のルーティングトラフィックで飽和させることなしに，ネットワークのトポロジー変化により速く適応することができる．リンク状態アルゴリズムでは，自分の隣接ノードに世界の状態を知らせるのでなく，ネットワーク全体に自分の隣接ノードの状態を知らせる．言いかえれば，1 つのルーティング領域中のすべてのノードが特定のリンクの状態を知っているので，各ルータは隣接ルータからの情報（多分古い）に頼らずに，自分で最適な経路を計算することができる．リンクの状態が用いられているので，目的地まで 1 つ以上の最小コストの経路が存在して，同一コストの複数の経路のルーティングを行うことが可能である．

1 つのインターフェイス上の各種のサービスに対して 16 ビットの指標が管理されている．これは必ずしも各サービスに対して異なる指標を使うという意味ではないが，少なくともより洗練されたルーティング制御を可能にする構造を提供している．

IS-IS（OSI における等価なプロトコル）と共に，OSPF は多くのリンク状態ルーティングプロトコルに共通な特徴をもっている．まず，時間が経つと，すべてのルータはネットワークに対する同じ像（つまり，同じルーティングデータベース）をもつようになる．各ルータは完全なトポロジーをもっているので，ベクトルに基づいた環境で起こり得るような振動効果を起こすことなく，ルーティングテーブルを独立に作ることができる．

OSPF は領域(area)の概念に基づいている．ノードの集合（通常はネットワークの集合）は 1 つ

の領域にまとめられる．領域ルータはその領域に属するあらゆるノードを見つけることができる．これはそのルータが目的地に到着するためのルーティング情報をもっていればよく，必ずしも目的ノードと同じサブネットワークに存在しなくてもよい．

他のルータは領域間のルーティングを行う．もしある領域ルータが自分の領域外のアドレスを見つけた場合には，そのパケットを領域間ルータに渡す．領域間ルータは他の領域までの経路がわかっていて，そこでそのパケットを領域内ルータに渡す．非常に大きなネットワークを構成するにはルーティング問題を階層的な領域(domain)に分割するのは有効な手法である．

OSPF は外部から求められた経路（例えばEGP 経路）を，内部から求められた経路と独立に配布する．ここで，ルーティング領域(OSPF 領域)内にあるノードは領域外のノードより正確な情報をもっていると仮定する．

OSPF における基本的な操作は，ルータの隣接ノードの情報を定期的に送り出すことである．これはリンク状態広告(link state advertisement : LSA)と呼ばれ，ルータの領域内全体に配布される．LSA はデータベースが同期をとれていることを意味する．定期的に各ルータはデータベースにアルゴリズムを適用して，最短経路のツリー(tree)を計算する．その後この表が転送データベース（ルーティングテーブル）を構成するのに使われる．

マルチアクセスネットワーク（例えば，イーサネット）では，指名ルータと予備の指名ルータが用意される．放送型のネットワーク上にあるルータはそれぞれLSA をその指名ルータに送り，そのルータが今度はネットワークを代表して LSA を生成する．

隣接ノードを発見するにはOSPF のハロープロトコル(Hello Protocol)が使われる．あるルータが“挨拶”（hello）を送るとき，パケットに最近“挨拶”パケットをもらったルータの一覧表を含める．これで，そのルータが隣接ノードの“挨拶”パケットを調べることによって，自分の出したパケットが隣接ノードに届いたことが確認できる．

放送型ネットワークでは，“挨拶”はマルチキャストとして送り出される．これは，イーサネットのようなネットワークでは隣接ノードは動的に獲得されることを意味している．ハロープロトコルはまた指名ルータを選出するのにも使われている．指名ルータが選出されると，そのルータと他のルータとの間には隣接性が存在する．ポイント対ポイント(point-to-point)リンクでは，隣接性は常に存在する．

データベースの同期制御は隣接ルータによって行われる．ルータは互いにリンク状態パケットのデータベース概略を送る．これで，各ルータは自分が情報をもらっていないルータに対する要求を作ることができる．

2つの領域に属するルータは2つのデータベースを保持する．つまり，境界にあるルータはより重要な役割を必要とされている．領域はサブネットの概念の一般化されたもののようなものである．サブネットは領域の一部であり，領域はネットワーク（ルーティング領域とも呼ばれる）の一部である．

1つの領域の中でのルーティングはOSPFルーティング階層の下位レベルで行われる．各領域はOSPFバックボーンと呼ばれるASの脊柱(spine)を通して接続されている．バックボーンは基本的には複数の領域に属するルータ（境界ルータ）の集まりである．

OSPF階層（1つ目は領域内，2つ目はASのバックボーン上）の最上位レベルはAS内のルーティングである．このレベルでは，EGPあるいは境界ゲートウェイプロトコル(Border Gateway Protocol : BGP)のようなプロトコルが経路情報の交換に使われている．その後，外部経路情報はASの全領域に配布される．

3.12 外部ゲートウェイプロトコル

外部ゲートウェイプロトコル(EGP)は2つのルータ領域の境界にある隣接ノードが，互いに到達可能情報を交換するために使われる方法の1つである．これは1つの自律システムの中でも用いられるが，普通は2つの管理領域の間でルーティング情報を交換するために使われている．

EGPは等位のノード(peer)の間で交換されるような一連の基本的なメッセージを定義している（図3-22および3-23を参照)．EGPプロトコルは3つの部分から構成されている．

・隣接ノードの獲得

・隣接ノードの到達可能性の監視

・ネットの到達可能性情報の交換

このプロトコルは，HelloとI-Heard-You(IHU)のメッセージ交換を用いた周期的な状態監視(polling)に基づいている．この他に隣接ノードに対する周期的問合せや，更新要求のコマンドももっている．

すべてのEGPのコマンドおよび応答には，順序番号が付けられている．ある特定の隣接ノードからのコマンドに付けられた最後の順序番号が，別の順序番号を受信するまでの間，その隣接ノードに対するすべての応答および表示に使われる．順序番号の制御は実装に依存するが，各隣接ノードに対して異なる番号を用い，pollコマンドの直前にその値を増加する方法が推奨されている．

EGPはインターネットを核(core)ASといくつかのスタブ(stub)ASに分ける．核ASはスタブの間でルーティング情報を交換するハブの役割を果たす．典型的には，NSFnetの核から見た場合，地域ネットワーク(regional network)はスタブである．これに対して，(EGPを用いる）ローカルネットワークがスタブに，地域ネットワークが核となる．

隣接ノードの習得は双方向のハンドシェーク方式に基づいていて，これに関してはEGPが要求および確認メッセージを交換するのにまかせなければならない．獲得作業は停止と停止確認メッセージによって終了される．

隣接ノードへの到着可能性はHelloとIHUの応答によって決定される．ハローを送ったゲート

ウェイは能動モード(active mode)であり，応答したゲートウェイは受動モード(passive mode)である．それ以降は，ネットワークの到着性は問合せと更新によって判断される．ネットワークがEGPパケットによって飽和してしまわないように，メッセージどうしの間に最小2分間の間隔がおかれる．

典型的なルータは，EGPの他に内部ゲートウェイのルーティングプロトコルも使う．1つの典型的な例として，バークレー4.2 EGPの実装では，ルータは2つの表をもっている．1つにはEGP交換によって習得した外部経路が記載されている．2つ目の表には，その他のプロトコル(RIPなど）で習得した内部経路が記載されている．

ある新しいEGP更新情報が得られたら，その情報は外部ルーティングテーブルに記録される．一般的な規則として，助言を出したゲートウェイが現在のゲートウェイと同じならパケットを更新し，また別のゲートウェイがより低い指標を助言したなら更新を行う．一定時間たつと，更新されていない経路は除去される．一般的には最大問合せ間隔(maximum poll interval)の3倍の時間更新されない経路は除去される．

ここで，デフォルト経路は例外である．多くのルーティングプロトコルでは，あるネットワークが完全に未知な場合に使われるデフォルト経路を保持している．その基本的な考え方としては，このルータは目的地がわからなくても，他のより大きくて賢いルータがわかるかもしれないということである．

EGPメッセージタイプ		
Request		隣接ノードの獲得および/または問合せ変数の初期化を要求する
Confirm		隣接ノードの獲得そして/または問合せ変数の初期化を確定する
Refuse		隣接ノードの獲得を拒否する
Cease		隣接ノード消去を要求する
Cease-ACK		
Hello		隣接ノードの到達可能性を要求する
I-H-U		隣接ノードの到達可能性を確認する
Poll		網の到達可能情報の更新を要求する
Update		網の到達可能性情報を更新する
Error		
固定パラメータ		
P1	30秒	相次ぐhelloコマンドの最小間隔
P2	2分	相次ぐpollコマンドの最小間隔
P3	30秒	リクエストやシースコマンドの再伝送の最小間隔
P4	1時間	アップおよびダウンの状態において命令や応答がない場合状態変数が保たれる時間
P5	2分間	獲得と消去状態において状態変数が保たれる時間
P4とP5は中止タイマが実装されたときだけ使われる．		

3-22 外部ゲートウェイプロトコル

EGPヘッダのフォーマット		
EGPのバージョン番号	1バイト	現在＝2
タイプフィールド	1バイト	
コード	1バイト	メッセージのサブタイプ
状態	1バイト	メッセージに依存する状態情報
誤り検出	2バイト	
自律システム #	2バイト	
順序 #		
変数フィールド		
隣接ノード獲得メッセージ		
タイプ		タイプ＝3
コード		0：要求命令
		1：応答確認
		2：応答拒否
		3：cease命令
		4：cease-ack命令
状態フィールド		0：未指定
		1：能動モード
		2：受動モード
		3：リソース不足
		4：管理上に禁止されている
		5：ダウンになる
		6：パラメータ問題
		7：プロトコル違反
hello間隔	2バイト	秒単位
問合せ(poll)間隔	2バイト	秒単位
隣接ノードの到達可能メッセージ		
タイプ		タイプ=5
コード		0: hello命令
		1: IHU応答
状態		1: アップ状態
		2: ダウン状態

3-23 EGPパケット形式

問合せコマンド		
タイプ		タイプ＝2
コード		コード＝0
状態		0: 不明確
		1: アップ状態
		2: ダウン状態
IP発信元ネットワーク	1-3バイト	
アップデート応答/表示		
タイプ		タイプ＝1
コード		コード＝0
状態		0：不明確
		1：アップ状態
		2：ダウン状態
		128：自発的なメッセージビット
内部ゲートウェイの数	1バイト	
外部ゲートウェイの数	1バイト	
IPソースネットワーク		
ゲートウェイ 1		IPアドレス（ネットワーク番号なし）
距離の数		
距離1		
ネットの数		
ネット1,1,1		
ネット1,1,2		
距離2		
ネットの数		
ネット1,2,1		
ネット1,2,2		
ゲートウェイ N		IPアドレス（ネットワーク番号なし）
距離の数		
記述		各ネットワークのフィールドの記述
ゲートウェイの数		このメッセージにある内部/外部ゲートウェイの数
IPソースアドレス		われわれが到達可能情報を提供しているIPネットワーク番号(1-3octets)
ゲートウェイIPアドレス		ネットワーク番号なし(1-3octets)
距離		自律システムの構造による
ネット		このゲートウェイを通して到達できるIPネットワーク番号

3-23（続き）EGPパケット形式

誤り応答/表示		
タイプ		タイプ=8
コード		コード=0
状態		0：不明確
		1：アップ状態
		2：ダウン状態
		128：自発的なメッセージビット
変数フィールド	2バイト	
理由		0：未指定
		1：EGPヘッダのフォーマットが間違っている
		2：EGPデータフィールドのフォーマットが間違っている
		3：到達可能情報がない
		4：選出のレートが高すぎる
		5：応答がない
不良データ		この誤りが発生したEGPヘッダの3つの32ビット長語

3-23（続き） EGPパケット形式

最悪の場合，隣接のルータも目的地がわからないので，ICMP の目的到着不可能メッセージを返す．少しましな場合には ICMP からの再方向付け(redirect)メッセージが得られる．もっといい場合は，その隣接のルータは確実に目的地を知っており，そのパケットを正しい方向に送り出す．

3.13 NSFnet と Milnet との間のルーティング

異なる自律システムの間のルーティングの興味深い事例研究として，Internet の中心である NSFnet と 2 つの軍事用ネットワーク（Milnet と ARPANET）との接続例がある．

NSFnet は中間レベルの地域ネットワークを接続するバックボーンネットワークである．これらの地域ネットワークは，今度は自律的に管理されている大学や研究センターのネットワークをつないでいる．

Milnet と ARPANET の 2 つの軍事用ネットワークは，国防省のいろいろな秘密ではない用途に使われている．しかし，Internet と ARPANET 研究センターとの間には多くの関わりがある．

1989 年中頃までには，中間レベルネットワークには 3 つのゲートウェイが存在した．これらのゲートウェイは NSFnetNSS のコアルータとの間に 1 つの EGP セッションを保持し，Defense Data Network(DDN) Mailbridge の間にもう 1 つのセッションを保持していた．

DDN から NSFnet への内向きのトラフィックの処理は，各 NSS が DDN の広告した経路を異なる指標で受けることによって行われていた．これにより 1 つの NSS が DDN に対して“ホット”な経路となり，その他の 2 つ（より高い指標をもつもの）はバックアップとなっていた．

NSFnet から DDN への外向きのトラフィックの場合は，各 NSS ルータは中間レベルネットワ

ークのゲートウェイと EGP 等位関係(EGP peer relationship)を保っていた．EGP を通して，NSFnet に接続されているネットワークの一部を等位ノードに対して宣言する．中間レベルネットワークにある NSFnet ゲートウェイは，もう 1 つの EGP セッションで自分と等位な DDN メイルブリッジにその情報を再配布していた．

1989 年中頃に，地域ネットワークにおけるゲートウェイの使用は直結ネットワークに置き換えられた．これはイーサネットを用いて NSS からメールブリッジまで連結できるように，メールブリッジにイーサネットのアダプタを置くことによって簡単に実現された．1989 年の 11 月には，このような 2 つのイーサネットに基づいたバックボーンが配置された．1 つは NASA-Ames 研究センター(カリフォルニア州 Moffet Field)に，もう 1 つは東海岸の Mitre(バージニア州 Reston)である．

このような高能力リンクによって，よりよい負荷分散を行うことができた．1 つのゲートウェイをホットリンク(hot link)にする代わりに，NSFnet はすべての動作しているゲートウェイに負荷を分散することにした．これは特定のネットワーク番号が常にある特定のメイルブリッジを優先するようにすることで実現された．動作休止の場合，他のものがその負荷を引き継ぐ．

NSFnet はこれらの DDN リンクをそれぞれ 1 つの異なった自律システムとして扱っている．NASA を通して習得した経路は AS 164 と名づけられ，Mitre から習得した経路は AS 184 と名づけられる．

NSFnet の内向きのルーティングは分散された方策のルーティングデータベースに基づいている．各ネットワーク番号はある正当なネットワークの一覧表と比較して，その正当性を確認する．各サイトにおいてその番号にある特定な指標を関連させる．

例えば，10 番のネットワークは ARPANET である．Palo Alto の NSS はそのネットワーク番号を習得して，それに 10 の指標を付ける．Maryland の College Park にある NSS もこのネットワークを習得して，それに 12 の指標を付ける．この時点で，DDN ネットワークは 2 つのブリッジの間でランダムに分割されている．最終的には，より“近い”ものを検出して，各 DDN ネットワークに対してそれに最も近い NSS を割り当てる．

NSFnet がある特定なネットワークの DDN に指標を決めるときは，問題になっているトラフィックをバックボーンに挿入する NSS とゲートウェイ NSS との間の距離に基づいて行う．これは基本的にはホップ数の統計である．

例えば，Princeton にある NSS は Mitre にある NSS から 1 ホップで，NASA-Ames からは 3 ホップである．このため，Mitre Mailbridge は NASA-Ames メールブリッジと比べて，よりよい指標がもらえる．そこで，Princeton に情報を送りたいネットワークは，まず Mitre メールブリッジに送って，それが Mitre の NSS へ行ってから Princeton の NSS へ行って，それから出てくる．

NSFnet を経由した DDN への接続は中間レベルネットワークを通して行われ，それが NSFnet に知らされる．あるネットワークが DDN に直接のコネクションをもっていたとしても，NSFnet

に基づく経路をバックアップとして望むかも知れない．この場合，そのメールブリッジは NSFnet の NSS から非常に大きな指標を受け取るであろう．

ここまでの議論は何かしら簡略化された環境であることに注意されたい．もし NSFnet のバックボーンを経由して，DDN-DDN 通信を行いたいとしたらどのようにすればよいだろうか．NSFnet は DDN より速いので，これを行うことは意味がある．この場合，RFC で見るような非常に限られた接続点しかない場合でなく，より一般的な網状構造になろう．

3.14 Sun 上の TCP

Sun 上の TCP モジュールはシステム起動の時点で始動される．一般には，/etc/rc.boot のファイルに，設定するための一連のコマンドが入っている．ifconfig 命令はシステム中の各ネットワークインターフェイスのパラメータを設定する．

```
ifconfig ie0 hostname netmask + -trailers up
```

この例では，ifconfig コマンドはある特定のインターフェイス（例えば，ie0 は最初のイーサネットカード）が特定のホスト名をもっていることを示している．ネットワークマスクは＋で指定されていて，これはネットワーク情報サービス(NIS)のマップ（例えば，netmasks. byaddr）を参照することを意味している．

-trailers は，そのインターフェイスが標準イーサネット形式を使うことを意味している．Trailers はバークレー Unix システムで使われているヘッダ情報をパケットの最後に置く方法である．なぜこのようにするか？ これはデータが到着したとき，それをメモリの境界に整列(align)しておくことができるからである．もしデータが整列されていなければ，これらをユーザに供給する前にバッファにコピーしなければならない．Trailers は厳密なバークレー環境でのみ動作するので，一般的なイーサネットのトラフィックでは混乱が起こる．

最後に，ifconfig 命令はネットワークインターフェイスを立ち上げる．その上で，そのカードの放送アドレスを設定して，その他のパラメータを制御するために（例えば，そのインターフェイスが非公開(private)であることを指定する)，再度このコマンドが発行される．

もちろん，もしこれがディスクレスノード(diskless node)であるなら，イーサネットを初期化す

hosts	ホスト名とIPアドレス
ethers	ホスト名とそのイーサネットの番号
networks	ネットワーク名と番号
protocols	IPプロトコル名と番号
services	IPサービス名とそのポート番号
netmasks	ネットワーク名とそのネットマスク

3-24 /etc 設定ファイル群

```
# /etc/hosts file
# local net 1: 192.9.200: 10mbps Ethernet: Engineering
192.9.200.1     jekyll          # me
192.9.200.2     usher
192.9.200.3     raven
# local net 2: 192.9.201: Marketing
192.9.201.11    flugg
192.9.201.12    fluff
192.9.201.4     jeckyll-hyde    # me again
```

3-25 /etc/hostsファイル

る必要はない．その作業はディスクレスノードが起動する前に行われるからである．すなわちオペレーティングシステムを見つけるためにネットワークが必要である．

/etc/rc.boot 中のコマンドの他に，システム中には情報を格納しているいろいろな設定ファイルが存在する(図 3-24 を参照)．システムの数が多くなると，これらの設定ファイルを管理するのが困難になる．第 9 章では NIS を使ってこれらの設定ファイルを置き換える方法を述べる．

重要な設定ファイルの 1 つはホストのデータベースである．このファイル中には IP アドレスとホスト名（あるいはニックネーム(nickname)や別名(alias)）との間のマッピング情報が記録されている．これによってユーザはサーバを指定するのに IP アドレスを記憶するのではなく，ホスト名を用いることができる（図 3-25 を参照)．

同様に，networks ファイルには，ネットワーク名の情報が格納されている．

ネットワーク名	ネットワーク番号	ニックネーム
ARPANET	10	arpa
Eng	193.9.0	
Corp	193.9.1	
Prog	193.9.2	

設定ファイルが準備できると，UDP あるいは TCP のポートで利用可能なサーバを表すしかるべきデーモンを起動する準備が必要になる．これらのデーモンの中には，常に走っているプログラムも含まれている．例えば，RPC 登録デーモンは到着した RPC 呼出しに応答するために常に走っている．その他のデーモンはメモリに常駐する必要のないプログラムである．これらのサービスを常に動かす代わりに，Internet サービスデーモン(inetd)と呼ばれるデーモンが起動される．このデーモンは必要に応じてサービスを呼び出す．

inetd.conf という設定ファイルにはいろいろな使用可能なプログラムが格納されていて，それらが使うポートが指定されている．デーモンはこれらのポートを listen する．FTP と Telnet は，このように動作するプログラムの典型的な例である．

TCP ネットワーク上のワークステーションの現状を監視する簡単な方法は，netstat コマンドである．図 3-26 と 3-27 に TCP/IP ネットワーク上の Sun ワークステーションに関するいろいろな

```
hydramatic% netstat -i
 Name  Mtu  Net/Dest   Address      Ipkts    Ierrs  Opkts   Oerrs  Col
 le0   1500 b14-spd-c  hydramatic   2729714  1      8479    1      208
lo0    1536 loopback   localhost    266      0      266     0      0
hydramatic% netstat -rs
routing:
        0 bad routing redirects
        0 dynamically created routes
        0 new gateways due to redirects
        0 destinations found unreachable
        0 uses of a wildcard route

hydramatic% netstat -r
Routing tables
Destination          Gateway      Flags  Refcnt  Use   Interface
localhos             localhost    UH     1       92    lo0
129.144.200.0        commserv     UG     0       0     le0
swan-cent            commserv     UG     0       0     le0
b5-spd-perf-ddlab    commserv     UG     0       0     le0
envtest-lab          commserv     UG     0       0     le0
b21-gpd-net-1        commserv     UG     0       0     le0
b10-springnet        commserv     UG     0       0     le0
b14-spd-fddilab-b    bodacious    UG     0       0     le0
analoglab            commserv     UG     0       0     le0
b17-dsd-hw-office-a  commserv     UG     0       0     le0
b8-gpd3              commserv     UG     0       0     le0
b1-labnet            commserv     UG     0       0     le0
b19-mps-sim          commserv     UG     0       0     le0
b16-wsg-c            commserv     UG     0       0     le0
b14-spd-a            bodacious    UG     4       396   le0
b14-ta-users         bodacious    UG     0       0     le0
fddi-net             commserv     UG     0       0     le0
b1-spd-dae-lab       commserv     UG     0       0     le0
b15-wsd-sst          rats         UG     0       0     le0
cte-net              commserv     UG     0       0     le0
b19-disknet          rats         UG     0       0     le0
b21-gpd-net-2        commserv     UG     0       0     le0
b18-ssd-bb           commserv     UG     0       0     le0
galaxy-net           commserv     UG     0       0     le0
b17-dsd-hw-office-b  commserv     UG     0       0     le0
b15-labnet           commserv     UG     0       0     le0
b8-gpd4              commserv     UG     0       0     le0
mtnview              commserv     UG     0       1     le0
b19-vlsi-circuit     commserv     UG     0       0     le0
b16-wsg-d            commserv     UG     0       0     le0
b19-sundragon-hwl    commserv     UG     0       0     le0
```

3-26 ルーティングテーブルの状態

```
hydramatic% netstat -s
  udp:
    0 incomplete headers
    0 bad data length fields
    0 bad checksums
    0 socket overflows
  tcp:
    2169 packets sent
            844 data packets (11571 bytes)
            0 data packets (0 bytes) retransmitted
            492 ack-only packets (415 delayed)
            3 URG only packets
            10 window probe packets
            747 window update packets
            73 control packets
    5873 packets received
            904 acks (for 11638 bytes)
            25 duplicate acks
            0 acks for unsent data
            5135 packets (6052774 bytes) received in-sequence
            2 completely duplicate packets (2920 bytes)
            0 packets with some dup. data (0 bytes duped)
            177 out-of-order packets (229988 bytes)
            0 packets (0 bytes) of data after window
            0 window probes
            2 window update packets
            1 packet received after close
            0 discarded for bad checksums
            0 discarded for bad header offset fields
            0 discarded because packet too short
    23 connection requests
    19 connection accepts
    38 connections established (including accepts)
    44 connections closed (including 0 drops)
    4 embryonic connections dropped
    904 segments updated rtt (of 927 attempts)
    12 retransmit timeouts
            0 connections dropped by rexmit timeout
    0 persist timeouts
    3 keepalive timeouts
            0 keepalive probes sent
```

3-27 Netstat 命令

```
                3 connections dropped by keepalive
icmp:
   3 calls to icmp_error
   0 errors not generated 'cuz old message too short
   0 errors not generated 'cuz old message was icmp
   Output histogram:
                echo reply: 22
                destination unreachable: 3
   0 messages with bad code fields
   0 messages  minimum length
   0 bad checksums
   0 messages with bad length
   Input histogram:
                destination unreachable: 17
                echo: 22
                #9: 1214
   22 message responses generated
ip:
   2446450 total packets received
   0 bad header checksums
   0 with size smaller than minimum
   0 with data size  data length
   0 with header length  data size
   0 with data length  header length
   24 fragments received
   0 fragments dropped (dup or out of space)
   0 fragments dropped after timeout
   0 packets forwarded
   6 packets not forwardable
   0 redirects sent
   0 ip input queue drops
```

3-27（続き） Netstat 命令

情報を示す．ここで，各ワークステーションにルーティングテーブルがあり，プロトコルの動作が可能であることに注意せよ．これらの情報の使い方はネットワーク管理者によって異なる．ある管理者は状況が悪くなると，単に netstat とタイプする．より大きな環境では，netstat に管理されるような情報は SunNet Manager を使ってアクセスされる．これは 15 章で述べられる．

SunNet Manager では，ネットワーク管理者はある事象に対して関心があると通知する．例えば，図 3-26 では重複パケット数があるレベルに達したときは通知してほしいとしている．例では，その数は 5783 の中の 2 であり，おおよそ全体の約 0.04%である．重複の数が 1%に達したら，アラーム音を出してもらってもよい．

ここで，このような監視作業を行う代価のために，ネットワーク管理者は問合せする対象をより厳しく選ぶようになる．すなわち変数を監視するごとに，あるプログラムがシステムを定期的

に問合せなければならない．これによって，少なくはあるがゼロではない資源が使われてしまう．

3.15　コネクションの起動

この時点で，少し前に戻って，あるユーザがネットワーク上にあるコネクションを起動するときどういうことが起こるかを調べてみる価値がある．ある TCP 期間が起動されるとき起こることについては，Greg Marshall の ConneXions に面白く記述してある．次に述べるのはその記述の凝縮である．

ここでは，ユーザがカリフォルニア大学バークレー校の化学科から Telnet セッションを起動しようとしていると仮定しよう．

```
myhost% telnet violet.cchem.berkeley.edu
```

ここで，ユーザは求めるホストの名前があらかじめわかっていると仮定する．このような情報は多くの場合，端末の側の壁に貼ってある PostIt のメモに載っているが，X.500 のディレクトリからも得られる．

Telnet のプログラムはドメイン名前システム(Domain Name System)を使って IP アドレスを探し出す．この情報はすでに自ホスト内のネームキャッシュ(name cache)に存在するかも知れない．そうでなく，またわれわれが edu 領域外にあるなら，Telnet はネームサーバに問い始める．edu のルートに着くと，アドレスは知らないが Berkeley. Edu へのポインタが与えられる．そのうちクライアント上のネームリソルバ(name resolver)はある IP アドレスを得る(ネームリソルバはおそらく Telnet コマンドの一部分であるか，あるいはそれにリンクされている)．そこで，Telnet は周知のソケット番号を探し出す必要がある．バークレーの Unix オペレーティングシステムでは，多くの場合/etc/services を調べる．

この時点では，Telnet は自ホストの TCP に 1 つのソケットを求める．そのソケットを獲得すると，宛先アドレスと TCP ポートを渡してコネクションを設立しようとする．自ホスト側では，TCP は 1 つのポート番号を与える．他の方法として，ストリームを用いて TCP モジュールに接続させることもできる．

この時点で，2 つの TCP モジュールは同期ビットを交換することによってコネクションを設立する必要がある．ここで，SYN 交換を行うとき，以前のコネクションからの古いパケットにだまされないように，適切な順序番号を選ぶ必要がある．

SYN パケットが用意できると，それは IP に渡され，そこでチェックサムが付加されて相手側に送られる．ここで，相手側は closed listening モードから SYN sent で，listening モードに遷移することになる．

パケットのルーティングは中間のゲートウェイを経由するかも知れない．このときサブネットのアドレスマスクが調べられて，そのアドレスが同一サブネットワーク内であるか否かが調べら

れる．もし同一サブネットワーク内ではないなら，ゲートウェイを使う（再方向づけを受け取ることになるかも知れない）．そこで同一サブネットワーク内のゲートウェイのIPアドレスを（ARPを通じて）解決し，伝送のためにネットワークに渡す．

より進んだ勉強のために

David D. Clark, *Fault Isolation and Recovery*, RFC 816, July, 1982.

———, *IP Datagram Reassembly Algorithms*, RFC 815, July 1982.

———, *Modularity and Efficiency in Protocol Implementation*, RFC 817, July, 1982.

———, *A Subnetwork Addressing Scheme*, RFC 932, January, 1985.

———, *Window and Acknowledgment Strategy in TCP*, RFC 813, July, 1982.

Comer, Douglas, *Internetworking with TCP/IP* (vol. 1, 2nd ed.). Prentice Hall (Englewood Cliffs, N.J.: 1991). TCP/IPに関する標準的なテキスト．

C. Hedrick, *Routing Information Protocol*, RFC 1058, June, 1988.

Jacobson, V., *Compressing TCP/IP headers for low-speed serial links*. RFC 1144, February 1990, 43 pp. (Format: TXT=120959, PS=534729 bytes)

Jacobson, V., Braden, R.T., and Zhang, L., *TCP extension for high-speed paths*. RFC 1185, October 1990, 21 pp. (Format: TXT=49508 bytes).

D.L. Mills, *Exterior Gateway Protocol Formal Specification*, RFC 904, April, 1984.

Nicholson, Andy, et. al., "High Speed Networking at Cray" *Computer Communication Review*, p. 99, v. 21, no. 1, January 1991.

David C. Plummer, *An Ethernet Address Resolution Protocol*, RFC 826, November, 1982.

J. Postel, *Internet Control Message Protocol*, RFC 792, September, 1981.

———, *Internet Protocol*, September, 1981, RFC 791.

———, *Transmission Control Protocol*, RFC 793, September, 1981.

———, *User Datagram Protocol*, RFC 768, August, 1980

J. Rekhter, *EGP and Policy Based Routing in the New NSFnet Backbone*, RFC 1092, February, 1989.

Rose, Marshall, *The Open Book* Englewood Cliffs, N.J.: Prentice Hall, 1989. OSIに関する現在では最もよい本．

Marshall Rose, Dwight E. Cass, *ISO Transport Services on top of the TCP Version: 3*, RFC 1006, May, 1987

Sidhu, Gursharan S., et. al., *Inside AppleTalk*. Addison-Wesley (Reading, Mass: 1989).

Stevens, W. Richard, *Unix Network Programming*. Prentice Hall (Englewood Cliffs, N.J.: 1990). Unix オペレーティングシステムの観点から TCP/IP および UUCP を述べたもの．内部構造に関する貴重な記述である．

X/Open, *X/Open Portability Guide, Networking Services*. Prentice Hall (Englewood Cliffs, NJ: 1988).

J. Yu and H-W. Braun, *Routing between the NSFnet and the DDN*, RFC 1133, November, 1989.

Chapter 4

ストリーム

4.1 プロトコル階層とオペレーティングシステム

ストリーム(STREAMS)とは，Unix System V Release 3の一部分としてAT&Tにより定義されたサービスである．それは後に，Sun(BSD)やAT&Tの系統のいくつかの一般的なUnixの変種を統合したオペレーティングシステムであるUnix System V Release 4に採り入れられた．

ストリームは，一連のモジュールをつなぎ合わせ，その連鎖に沿ってメッセージを受け渡していく方法を提供するものである．典型的な例として，プロトコル階層を構成することがあげられる．メッセージ(プロトコルデータユニット)はプロトコル階層を下っていき，ストリームの末端にあるデバイスドライバを通って出ていく．また，ドライバから入ってくるメッセージは，それぞれのモジュールを通って上ってくる．

ストリームはまた，メッセージ受渡しの一般的な構造を提供する．メッセージの標準的なセットの定義は，トランスポートレベルインターフェイス(TLI)のように，インターフェイスの一部分となる．TLIは，ユーザプログラムがトランスポート層を使って通信するための，AT&Tが定義した共通のインターフェイスである．トランスポート層に限らず他の層にも，例えばデータリンク提供者インターフェイスのようなインターフェイスがある．

われわれは，主にトランスポートレベルインターフェイスに集中することにしよう．というのは，それがトランスポート独立RPC(TI-RPC)のメカニズムを提供する技術だからである．TI-RPCは，例えばTCPのようなある1つの環境で設計されたRPCアプリケーションが，UDPやTP4のような他の環境の上でも動作することを可能にするものである．

この章で議論されるストリームやその他のインターフェイスは，それそのものがネットワーク用のツールであるわけではない．それらは，ネットワーク上のあるいはローカルな資源をアクセスするための，オペレーティングシステム上の単なる標準インターフェイスにすぎない．標準インターフェイスはそれ単独で相互運用性(interoperability)を提供するものではないが，それを使うことにより，ベンダは互換性のある装置やソフトウェアモジュールを供給することが容易にな

る.

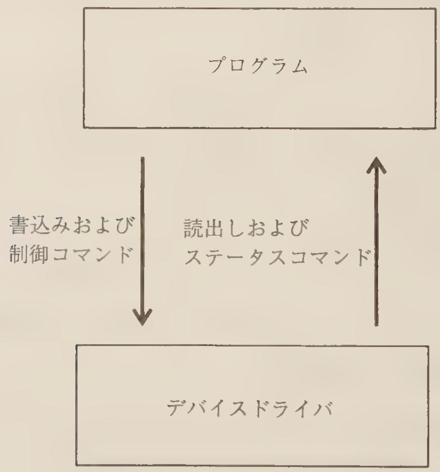
ストリームは，ネットワークサービスにアクセスするための1つの手段である．SVR4を採用する以前は，Sunオペレーティングシステムのネットワークインターフェイスはソケット（socket）と呼ばれる別の概念に基づいていた．ソケットは，単なるオペレーティングシステムに対する標準インターフェイスであり，TCPやUDPのポートをアプリケーションのためのアクセスポイントに対応づけるものであった．

ソケットは，XNS，TCP/IP，Unix内通信といったいくつかの異なった種類のネットワークのための，共通のアプリケーションプログラミングインターフェイス(API)として実装されてきた．ソケットライブラリはほとんどのUnixシステムに存在し，PC-NFSの一部でもある．しかしそれはストリームよりも原始的なサービスであり，トランスポート層でのプログラミングのセマンティクスに深く結びついている．SVR4では，ソケットライブラリは既存のBSDアプリケーションとの後方互換性を保つために，ストリームの上に提供されている．

4.2 キャラクタI/O機構

ストリームを理解するためには，同一計算機内での入出力処理の方法から始めるのがよいであろう．プログラムからアクセスされるデバイス，例えばモデムなどは，すべてデバイスドライバをもっている．デバイスドライバは，キャラクタを受け取ってそれをデバイスに送り出すこと——すなわち，モデムを通してデータを送り出すことができる(図4-1参照)．それぞれのデバイスドライバには一連のコマンドがあり，それによって行うべき動作を指定することができる．ストリームの基本にある考え方は，これらのデバイスすべてに対し，共通のインターフェイスを提供するということである．

ストリームの機構にはもう1つ重要な拡張がある——メッセージベースI/O機構(message-

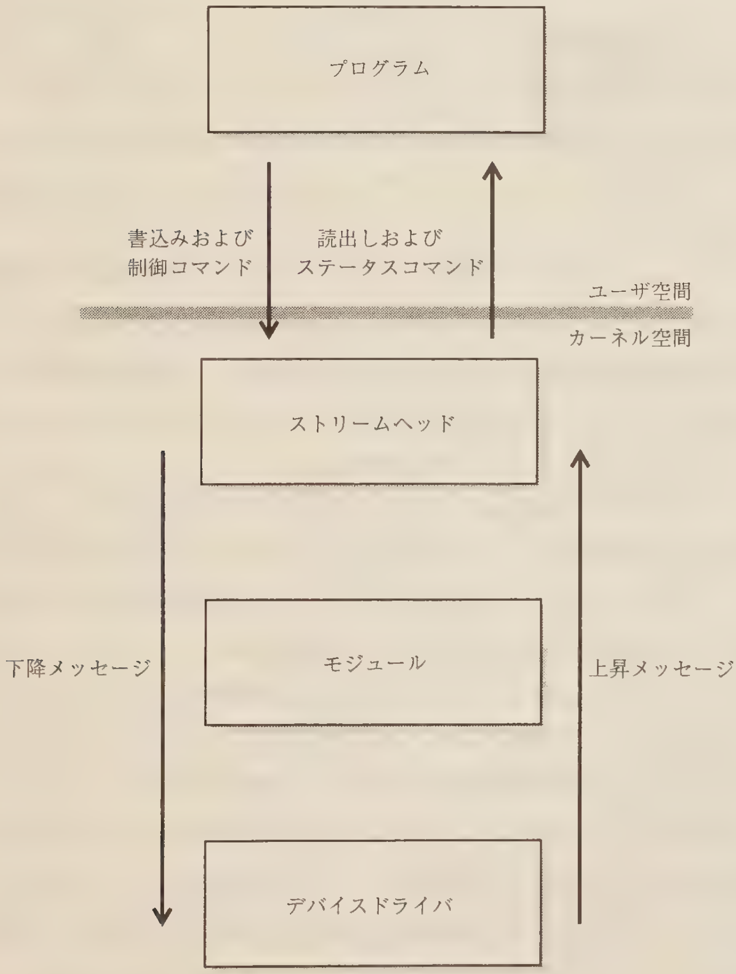


4-1 キャラクタI/O

based I/O mechanisms) のサポートである．伝統的なデバイスドライバは，キャラクタ I/O 機構に基づいている．ストリームでは，ユーザはメッセージを発送することができ，それはデバイスドライバの待ち行列(キュー)に入れられる．デバイスドライバは，時間ができるとそのメッセージを処理する．すなわち，ストリームはメッセージベースであり，それをキューイングすることができるのである．後で見るように，この機構は，同様の特徴をもつネットワーク環境において理想的な枠組みを提供する．

4.2.1 基本的なストリームの構成要素

ストリームは 3 種類の構成要素からなる．ストリームヘッド，モジュール，そしてデバイスドライバである(図 4-2)．モジュールはなくてもよいが，ストリームヘッドとドライバは必須である．ストリームヘッドがプログラムと直接交信するものであり，プログラマから見えるインターフェイスとしてデバイスドライバにとって代わる．



4-2 基本ストリームメカニズム

モジュールはストリームヘッドとドライバの間に位置する．LANの場合は，ドライバは例えばイーサネットドライバであろう．するとモジュールはTCPあるいはIPあるいはUDPのソフトウェア実装ということになる．

デバイスドライバは，ストリームと互換性のあるデバイスドライバならばどのようなものであってもよい．それは上からの，つまりストリームヘッドまたは何らかのモジュールからのメッセージを受け取る責任をもつ．そしてメッセージを受け取ると，それを翻訳し，制御しているデバイスに送り出す．ドライバがイーサネットドライバならば，データをLANに送り出すということである．また，ドライバはデータを受け取り，ストリームヘッドに向かって送り出す．その後ストリームヘッドはそれをユーザプログラムに渡す．

ファイルのように，ストリームヘッドは読出し／書込みのような基本的な操作を行うことができるし，それ以外にもいろいろなコマンドセットをサポートする．ストリームが最初にオープンされたとき，それはドライバとストリームヘッドからなる．

アプリケーションはストリームヘッドとドライバとの間のスタックにモジュールを積むことができる．プログラマは，ストリームの制御関数によって，モジュールをストリームに追加したり，ストリームから取り除いたりすることが可能である．モジュールをスタックに追加するには，プログラマはIOCTL関数の呼出しにおいて，望みのモジュール名をI_PUSH引数とともに与えてやればよい．最もストリームヘッドに近いモジュールを取り除くには，代わりにI_POPオプションが使えるであろう．さらに追加されるモジュールはストリームヘッドと一番上のモジュールとの間に順番に挿入される．アプリケーションがストリームヘッドにコマンド(例えばIOCTL)を送ると，ストリームヘッドはデータをメッセージの形にまとめる．メッセージは下流へ送られていく．ストリームヘッドの下流にある最初のモジュールはメッセージを検査し，何か動作を起こすべきかどうかを決める．モジュールはメッセージを変更したり，新しいメッセージを挿入したり，あるいは単に流れからメッセージを削除したりすることができる．

もしあるモジュールがメッセージを理解できなかったら，それは下流にある次のモジュールに渡される．次のモジュールはIOCTLの引数を吟味し，知っているタイプかどうかを決定する．例えば，われわれはトランスポートモジュールに対し，確認応答の来ないパケットに対して行うグローバルな再試行の上限を変更するようなIOCTLのコマンドを定義することができるであろう．理解できないメッセージが次々と受け渡されていくということは，既存のプログラムを壊すことなく新たな機能を追加できるということを意味する．メッセージがストリームの終わり――デバイスドライバ――に行き着いたときに，理解されないメッセージは捨てられる．

プログラムがストリームヘッドと交信する方法はオペレーティングシステムによって異なる．普通はストリームの動作を制御したり，伝統的なI/O機構をサポートしたり，さらにメッセージをストリームに送ったりストリームから受け取ったりすることができるような関数呼出しが提供される．

プログラマは，putmsg 関数呼出しを発行することができる．この関数はストリームヘッドに対して，メモリ上のある位置にあるデータが下流へ送られるべきものであることを知らせる．そしてストリームヘッドはユーザメモリからデータを得る．メッセージはコントロール部とデータ部の 2 つの部分からなる．データ部はメッセージの解釈者が使うものであり，コントロール部はメッセージの種別を示すものである．

プログラマが一連のプロトコルメッセージを生成するというシナリオを考えてみよう．プロトコルメッセージはストリーム中のモジュールによって解釈され，変更を受けて次のモジュールかドライバに送られる．例えば，ストリームの最上位は，セッションの確立やデータのやりとりのようないくつかのサービスプリミティブを提供するトランスポート層であるかもしれない．

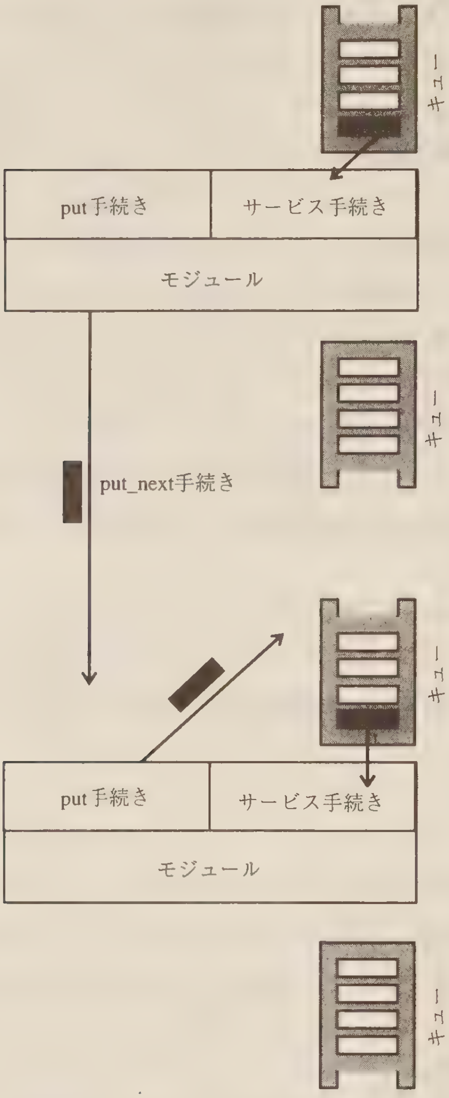
このモジュールは，プロトコルメッセージのコマンド部に含まれるコマンドのうちいくつかを認識するだろう．このモジュールが接続確立メッセージを認識すると，データ部をコピーする．おそらくこのメッセージのデータ部には，接続のために必要な遠隔ホストの宛先ネットワークアドレスなどが含まれているであろう．

するとトランスポートモジュールは，ネットワーク層に渡すべきメッセージを生成し，ネットワーク層ではルーティングヘッダを付加してメッセージをデータリンクドライバに渡すであろう．このようにして，それぞれのメッセージはプロトコル階層を下降していくに従ってヘッダを付け加えられていくことになる．

4.2.2 メッセージキューとフロー制御

ストリーム上の各構成要素はそれぞれ 2 つのメッセージキューをもっている．1 つは入力メッセージ用で，もう 1 つは出力メッセージ用である（図 4-3 参照）．これらのキューは，大量のメッセージがモジュールやドライバに殺到するのを防ぐためのフロー制御およびデータ優先配送サービス（expedited data service）の機構を提供してくれる．キューは，いくつかのメッセージを置いておくのに十分な広さをもった単なるメモリ上の領域である．モジュールはメッセージを下流に送りたいと思うと，put_next 手続きを起動する．put_next 手続きは次のモジュールの読出しキューにそのメッセージを入れる．

キューは，メッセージの一連のリンクと付随する制御データから構成される．キューの先頭は最初のメッセージを指しており，そのメッセージは次のメッセージへのアドレスをもっているといった具合いである．最後のメッセージ，つまりキューの末尾はキューの先頭へ戻るポインタをもっている．モジュールはメッセージの処理を終えると，それを put_next 手続きにより下流へ渡していく．put_next 手続きは，次のモジュールの入力キューに対応する put 手続きにそのメッセージを手渡す．put 手続きはメッセージを調べ，それがすぐに処理されるべきものかどうかを判断する．すぐに処理する必要のないものならば，キューに置くことができる．通常はメッセージはキューの末尾に置かれ，ポインタはキューの先頭を指すように調整される．



4-3 ストリームのキュー管理

モジュールは，対応するサービス手続きをもつことができる．このサービス手続きは定期的に呼び出され，その際にすべてのメッセージキューを空にしようと試みる責任をもつ．

ときには下流のモジュールの処理が滞ったために，上流のモジュールがメッセージを送れなくなることがある．このようなことが起こると，ストリームはブロックする．下流のキューがブロックされると，そのモジュールは自分のキューを空にすることができなくなり，下流のキューがブロックを解除しない限り，ゆくゆくは上流のすべてのモジュールが停止してしまうかもしれない．

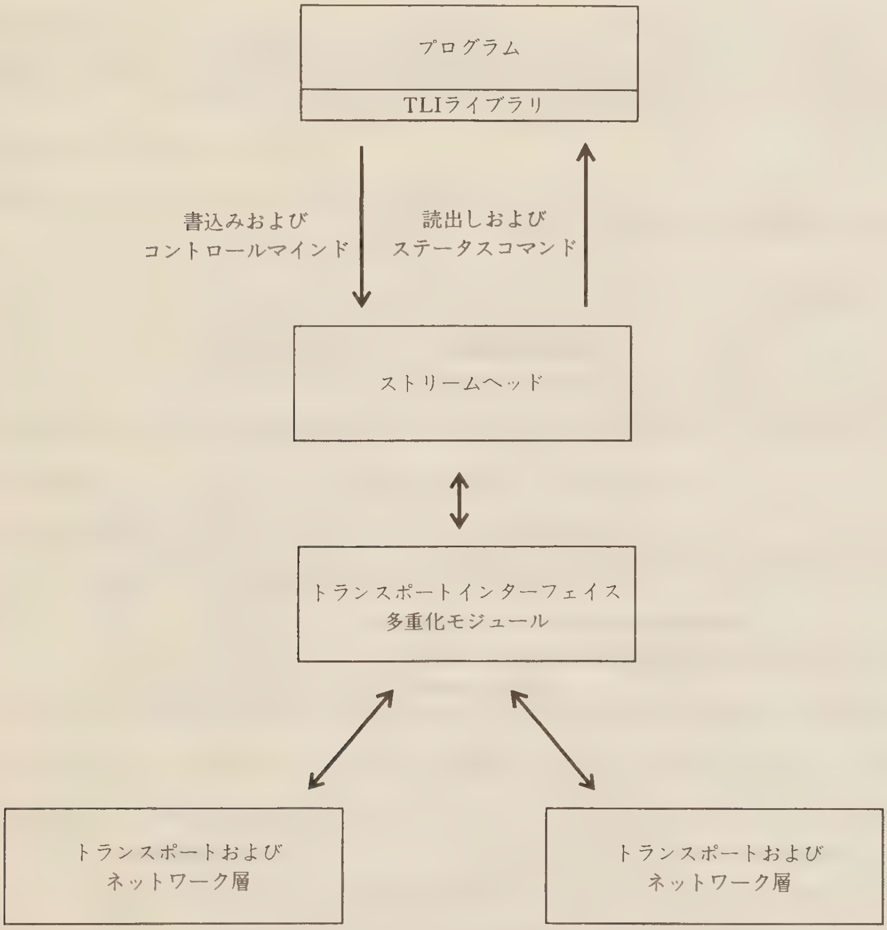
フロー制御はキューの長さに基づいて行われる．それぞれのキューには，キューごとに高水位線および低水位線が設定されている．上流のモジュールは，高水位線に達するまでキューにメッセージを追加することができる．ひとたびキューが高水位線まで達すると，低水位線に下がるま

で上流のモジュールはブロックされる.

通常，メッセージはキューの末尾に送られるが，高優先度メッセージとして知られるあるクラスのメッセージはフロー制御を受けず，キューの末尾に置かれることもない．その代わりに，それらのメッセージはキューの先頭の方で，より優先度の高いメッセージの後ろに追加される.

4.3 ストリーム多重化機構

基本的なストリームは, 1つの使用者と, 1つのドライバと, いくつかの(あるいはなくてもよい)モジュールとで構成される．しかし，多くの計算機環境では，1つのドライバをいくつかの使用者で共有しなければならない. ストリームは多重化することができ, いくつかの使用者が1つのストリームを共有することができる．多重化されたドライバは，重要なストリームのアプリケーショ



4-4 トランスポートレベルインターフェイス

ン，例えばトランスポートレベルインターフェイスに基づくものなどの，大切な構成要素である．

多重化されたストリームは，実際は2つ以上のストリームが束ねられたものである(図4-4参照)．多重化モジュール(multiplexing module)が，データを適切な上位または下位のストリームに送るために用いられる．多重化はどちらの方向に対してでもできる点に注意してほしい．いくつかの上位のストリームが多重化モジュールを共有することもできるし，いくつかの下位のストリームが多重化モジュールの制御下に置かれることもある．複数の下位のストリームを作るには，まず2つの別々のストリームをオープンする．最初のストリームはストリームヘッドと多重化モジュールから構成され，2番目のストリームはストリームヘッドとドライバから構成される．

次に，下位のストリームがI_LINKコマンドを用いて多重化ドライバに接続される．この接続により，下位のモジュールのストリームヘッドは，上に向かうすべてのデータを多重化モジュールに送るようになる．多重化モジュールからのデータはすべて下位のストリームヘッドに送られ，それからドライバに渡される．

I_LINKコマンドを用いて下位ストリームを追加することもできる．いくつかのTCPモジュールは1つのIPモジュールの上にサービスを多重化することができ，またIPモジュールはいくつかの異なるデバイスドライバのサービスを用いることができる．

4.4 トランスポートレベルインターフェイス

これまで見てきたように，ストリームはメッセージをモジュールからモジュールへ，そして最終的にはドライバへと送っていく一般的な機構を提供するものであった．モジュールがメッセージを認識できないときは，それはそのまま次に渡される．モジュールがメッセージを認識できた場合は何らかの処理を行って，通常は変更されたメッセージを下流に向かって送り出す．

ストリームはメッセージを受け渡すための機構を提供するが，メッセージが何であるかを規定するものではない．トランスポート提供者インターフェイス(Transport Provider Interface : TPI)が，トランスポート層サービスの異なる提供者が共通に認識すべきメッセージのクラスを定義している．トランスポートレベルインターフェイス(TLI)は，TPIに従うメッセージを生成するのに用いられる，標準のAPIである．

TPIに従うモジュールは，TLIインターフェイスを用いているどんなプログラムにもサービスを提供することができる．例えば，TCPで直接プログラムする代わりに，プログラマはTLIライブラリを用いてアプリケーションを書くことができる．実行時には，プログラムはOSIやTCP/IPのプロトコル階層を用いることができる．それはプログラムからは透明であるため，プログラムを書き直すことなしに下位のトランスポート機構を変更することが可能となる．

TLIにはもう1つ移植性を高めている要素がある．たくさんのベンダがトランスポートレベルインターフェイスを採用してきたため，少なくともプログラムのネットワーク関連部分は非常に

移植性が高い．もしプログラムがC言語の標準バージョンで書かれていれば，コードの残りの部分も移植可能になる．

TLIは2つの部分から構成される．TLIライブラリおよびTLIモジュールである．TLIライブラリはプログラムにリンクされ，支援ライブラリとなる．TLIライブラリは，プログラムの主要部分にあるTLI呼出しを受け取り，ストリームヘッドへメッセージを送るための適切な操作を行う．

一方，TLIモジュールはストリームが作られたときにオープンされる．そして，TLIライブラリからのリクエストを受け取り，そのうちのいくつかを標準TPIメッセージに変換する．TLIモジュールの下には1つまたは複数の異なるプロトコル階層が存在する．

TLIライブラリにより，どのような種類のトランスポート階層が用いられているかを気にすることなくネットワークベースのプログラムを書くことができるようになる．例えばプログラマは，どんなトランスポート階層の上でも，コネクションをオープンするにはt_connect関数呼出しを行えばよい．

また，接続要求を出してから接続が確立されるまで，待たなくてもよいようにすることも可能である．プログラムはすぐに他の処理を行い，あとから接続要求が成功したかどうかを調べるためにt_rcvconnectコールを行えばよい．

様々なTPIメッセージ(および対応する関数呼出し)が定義されている．例えばエラーハンドリングや優先データなどである．基本的な機能はプログラマが直接トランスポート層に対してプログラムを書いた場合と変わりはない．TLIのようなアプローチをとる利点は，透過性——すなわちプログラマがネットワークの種類を気にせずに，単一の手法でプログラムを書くことができるということである．なお，直接TLIを扱うようなプログラムが書かれることはほとんどない．通常，プログラマは，Netwise RPC Toolのようなより高いレベルの機構を用いるであろう．

より進んだ勉強のために

W. Richard Stevens, *Unix Network Programming,* Prentice Hall (Englewood Cliffs, N.J.: 1990).

(邦訳：「Unixネットワークプログラミング」篠田陽一訳，(株)トッパン)

Sun Microsystems, *STREAMS Programming Manual,* Part Number: 800-3826-10, Revision A of 27 March, 1990.

Chapter 5

RPC と XDR

5.1 なぜ RPC か?

これまでにサブネットワーク，ネットワーク，そしてネットワークのトランスポート層について述べた．トランスポート層に達することで，ネットワーク上の他のノードと送受信することができるようになる．TLI のような標準インターフェイスによって，多くの場合プログラマはどのトランスポート機構が(そしてどのネットワークとサブネットワークがその下に)使われているかについて気にする必要がない．

ファイル転送プロトコル(File Transfer Protocol : FTP)やその他多くの Internet のアプリケーションのように，トランスポート層をネットワークへのインターフェイスとしているアプリケーションもある．このようなアプリケーションは離れたインターネット(およびポート)アドレスに対してソケットを開き，データの読み書きを始める．単純なデータ転送にかかわる基本的な操作については，このネットワークのパラダイムで十分である．しかし，ネットワークを通じて 2 つのプログラムが通信しようとすると，他の様々な問題が生じ得る．

異機種ネットワークにおけるデータの表現法を例にとってみよう．整数やその他の多くのデータ型は，異なるコンピュータアーキテクチャではその表現法が違っている．あるマシンでは，整数は最上位ビットを MSB，最下位ビットを LSB として格納するが，別のマシンでは逆の流儀で格納されることもある．

分散アプリケーションを定義する際のもう 1 つの問題点は，離れたところにあるアプリケーションの部分と通信するための適切なパラダイムを見つけることである．ソケットと TLI インターフェイスのどちらも低レベルのネットワーク API を表現している．しかし，どちらも多くのアプリケーション開発者に親しみのあるプログラミングモデルを提供するものではない．

低レベルのインターフェイスは，この点については何もしないので，Sun は遠隔手続き呼出し(Remote Procedure Call : RPC)と外部データ表現(External Data Representation : XDR)と呼ばれる 2 つのプロトコルを付け加えた．遠隔手続き呼出しとは，2 つのプログラムが一連の要求，応

答，エラーメッセージを出すことのできる方法である．RPC を補足するものがプレゼンテーション層プロトコル XDR で，異機種ネットワークの間でこれらのメッセージの内容を解釈することができる．この 2 つのプロトコルは TCP/IP 環境におけるプロトコル階層の上位部分の事実上の標準になっている．

RPC と XDR の上にネットワーク情報サービス（Network Information Service : NIS）やネットワークファイルシステム（Network File System : NFS）のようなサービスが構築される．NIS については 9 章で議論するが，ネットワーク上のサービスや他の情報を得る方法の 1 つである．NFS については次章で議論するが，ファイルシステムの概念をネットワークにまで広げるものである．

これらのプロトコルの集合は Open Network Computing（ONC）という名前で出荷されている．しかし，プロトコルの集合全体に言及するための簡略形として NFS という名前がしばしば用いられる．ONC は 90 以上のベンダから販売されており，PC-DOS，VAX/VMS，すべての Unix の亜流，VM/CMS，Macintosh，MVS などを含む主要なオペレーティングシステム上で実装が利用可能である．

RPC/XDR の問題点については 2 つの段階に分けて述べよう．この章では，ネットワークで用いられるプロトコルについて考察する．通常このプロトコルは，アプリケーションプログラムにリンクされた RPC/XDR ライブラリを通してアクセスされる．プログラマーがネットワークを横断した呼出しを行うには，単にライブラリを呼び出すだけでよい．

RPC/XDR を直接プログラミングすることは，RPC/XDR を利用する第 1 の方法であり，多くの大規模アプリケーション，特に NFS はこの方法で書かれている．RPC ライブラリを使うことで，プログラマはネットワークアクセスの多くの問題を気にしなくてよくなる．しかし，それでもネットワークで何が起こっているかについていくらかの知識が必要とされる．

より高いレベルの抽象化を行うには RPC コンパイラを用いる．これはプログラマがいくつかのコンピュータを 1 つのシステムとして扱うことのできるような高級言語である．プログラマは入出力のある手続きを設計し，同一計算機上にある手続きと全く同じようにこの手続きを呼ぶ．

RPC コンパイラについては，アプリケーションの例を見た後に 7 章で議論する．流れを中断する理由は 2 つある．1 つには，基本的な RPC アプリケーションを見ることで，このパラダイムの能力を読者は理解できるだろうからである．ネットワークにまたがるプログラムをどうやって書くかに進む前に，なぜそのようなプログラムを書こうとするのかを見ておくことは有意義である．もう 1 つには，基本プロトコルの上によくできた RPC コンパイラを付け加えることで，非常に大きな価値を得ることができることがわかるだろうからである．基本プロトコルは，ネットワーク上でメッセージを送る手段である．洗練されたエラー処理やデバッギングのような便利なものは，この基本プロトコルの中には用意されていない．このような便利な機構は，RPC コンパイラによって付け加えられるコードの層を用いることによって提供されていることがしばしばある．

このようなわけで RPC プログラミングを 2 つのレベルで見ていこう．まず，ネットワーク上に

完結する情報，つまりプロトコルがある．次に，これらの呼出しが適切なサーバあるいはクライアントプログラムに渡される前になされる，ネットワークライブラリによる呼出しの翻訳がある．

5.1.1 プロトコル

RPC は非常に簡単なプロトコルである．すべてのメッセージは要求か応答のどちらかである．各メッセージは 32 ビットの転送 ID で区別され，要求と応答を関連づけるために用いられる．

これに加えて，要求でどのタスクが要求されているかを特定するために 3 つの 32 ビットの数がある．それらはプログラム番号とプログラムバージョンと手続き番号である．サーバプログラムの中には，コマンドの複数のバージョンをサポートするものもあり，ただ 1 つのバージョンしか受け付けないものもある．バージョン番号によってサービスの改良を上手に行うことができるようになる．

1 つの RPC 呼出しはいくつかの引数(argument)といくつかの戻り値とをもつことができる．しかし，戻り値の数を制限するコンパイラもある．例えば，rpcgen の初期の頃のバージョンでは 1 つの結果しか許さない(ただし，その結果は複雑なデータ構造であってもかまわない)．RPC Tool は複数の引数と戻り値をもつ手続きを受け付ける．引数のそれぞれは複雑な構造であってもよい．慣習として，番号ゼロの手続きは空の引き数をとり，空の戻り値をもっていて，プログラムを "ping" (存在の確認)できるようになっている．

インターフェイスに対してプログラム番号が割り当てられる(同じサーバプログラムに通信することで利用できる一連の関連した手続き)．ある意味で，プログラムの中の手続きのこの集合は，ネットワークライブラリを提供している．単一のサーバアプリケーションが，いくつかの異なったプログラム番号で異なるサービスを提供することもできる．プログラム番号はグループの中で割り当てられる．グループ 1 は Sun に管理されており，すべての商用のプログラムを含んでいる．グループ 2 は局所的な使用のためだけに限られる．グループ 3 は一時的なもので，コールバック RPC のために使われる．最後に，グループ 4 は将来の使用のために残されている．

RPC サービスは，TCP や UDP の周知のポートではなく，プログラム番号で識別されていることに注意しよう．rpcbind(ポートマッパとしても知られる)という特別なサービスが，あるサービスがその時点でどのアドレスを使っているかを離れたところのユーザに知らせるために用いられる．この柔軟性により，めったに使われないのに周知のポート番号を束縛することなく，多くの異なる RPC サービスが存在し得る．

小さなブートストラップ問題が存在する．あるアプリケーションがどのポートを使っているのかを見い出すための離れたサーバ上のポートマッパが，どのようにしたら見い出せるかという問題である．この問題を解決するために，ポートマッパだけは周知の RPC サービスとなる．つまり，ポートマッパは固定アドレスを用いる．

5.1.2　ライブラリへのアクセス

プログラマが利用できる基本 RPC 呼出しを図 5-1 に示す．これらの呼出しはいくつかのカテゴリーに分けることができる．最初の 2 つはクライアントあるいはサーバプログラムが利用できる呼出しである．その後にあるのはその他の呼出しであり，10 章でさらに詳細に述べる認証(Authentication)などがある．

図 5-2 ではネットワークを横断する典型的な RPC 呼出しを示している．(UDP または TCP のヘッダに埋め込まれた)RPC パケットにはトランザクション ID が含まれており，要求，応答，エラーを照合することができる．ここで呼び出されているプログラムは NFS(Network File System)である．手続き番号 4 が呼び出され，ファイル名を参照するために用いられている．

RPC モジュールにとって，呼出しの目的は取るに足りないことであることに注意しよう．RPC が行うことはサーバプログラム，この場合はサーバマシン上で動いている NFS デーモンに呼出しを伝えることである．

10 章で詳述するように，サーバプログラムは信任状(credentials)によって，誰がそのプログラムを使っているかを知ることができる．クライアントが認証されると，サーバはどのようなアクセ

RPCサーバコール
プロシージャのRPCサービスパッケージへの登録
RPC要求の到着待ちと適切なサービスの呼出し
一般的なトランスポート独立RPCサービスの生成
テスト用のRPCサービストランスポートの生成
TCPトランスポートを用いたRPCサービスの生成
UDPトランスポートに基づいたRPCサービスの生成
プログラム番号およびバージョン番号をサービスディスパッチ手続きに対応づける
RPC要求の引数の復号化
手続きを呼び出した側のネットワークアドレスの獲得
RPCの結果の返送
RPC/XDRによって割り当てられたデータを，引数を復号化するときに解放する
認証の不成功のためサービスの拒否
サービスがパラメータを解釈できないときに呼ばれる
サービスがこの手続きを実装していない
プログラムがRPCサービスに登録されていない
バージョンがRPCサービスに登録されていない
認証が十分でない
ディスパッチルーチンへのプログラムとバージョンのマッピングの除去
RPCサービス伝達ハンドルの破棄

5-1　RPCライブラリコール

RPCクライアント
離れたところにある手続きの呼出し
離れたところにある手続きの一斉呼出し
一般的なトランスポート独立RPCクライアントの生成
テスト用のRPCクライアントの生成
TCPトランスポートを用いたRPCクライアントの生成
UDPトランスポートを用いたRPCクライアントの生成
クライアント生成不成功の理由を示す大域変数
クライアントハンドル生成不成功の理由に関するメッセージを標準エラー出力に書く
クライアントハンドルに対応する離れたところにある手続きを呼び出す
エラー情報をクライアントハンドルからエラー構造体にコピーする
エラー番号に一致したメッセージを標準エラー出力に書く
クライアント呼出し(clnt_call)不成功の理由についてのメッセージを標準エラー出力に書く
RPC/XDRによって割り当てられたデータを，結果を復号化するときに解放する
クライアントのRPCハンドルの破棄
認証
RPC確証ハンドルを照合なしで返す
RPC確証ハンドルをUnixの承認を得て返す
デフォルトのUnix確証ハンドルを返す
Secure RPCへのRPC確証を返す
オペレーティングシステムに依存しないネット名をシステム内で使用できる
信任状 (credential) に変換する
認証情報ハンドルの破棄
Secure RPC
ホストあるいはドメイン依存のホスト名をオペレーティングシステムのネット名に変換する
あるユーザのネット名を獲得する
ネット名をホスト依存の名前に変換する
オペレーティングシステムに依存しないネット名をドメイン依存なユーザIDに変換する
ユーザIDをネット名に変換する
RPCBind
プログラムとアドレスの間のマッピングを確立する
ポートマッピングに対してRPCプログラムのリストを返す
サービスが待っているポート番号を返す
ポートマッパにRPC呼出しを行うよう知らせる
ポートへのマッピングを破棄する

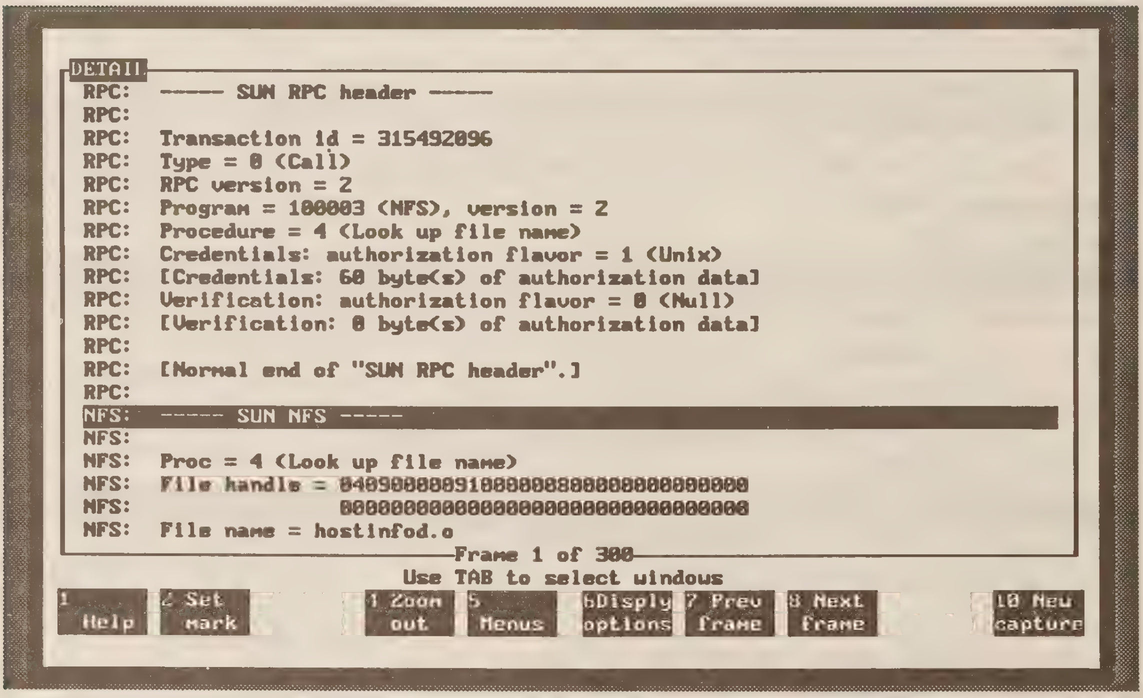
5-1(続き)　RPCライブラリコール

ス方法を提供するかを決めることができる．RPC における認証はいろいろな“流儀”で提供されている．図 5-2 では Unix 風であり，ユーザを認別するためにクライアントの UID が引き渡される．利用者が新しい流儀を付け加えることもできる．

図 5-3 はトランザクションが，今回は別のプログラムに送られるところを示している．この場合認証の流儀は 3 であり，公開鍵(public key)暗号に基づく Secure RPC 流である．

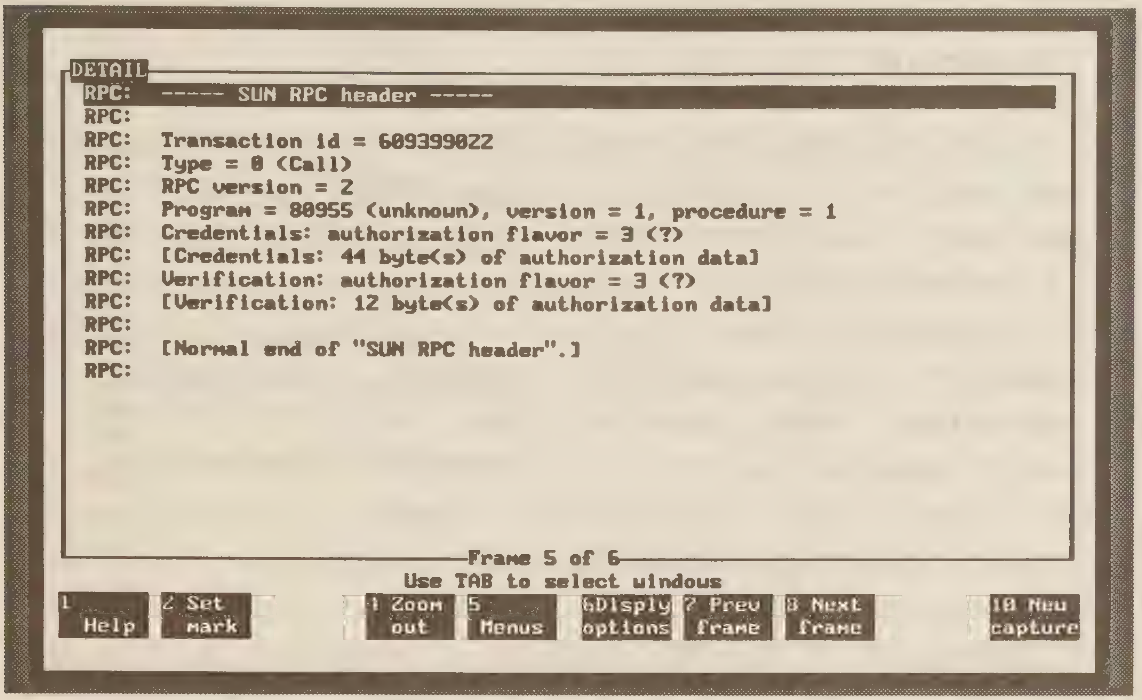
RPC が知っている流儀のそれぞれに関して，RPC は信任状を調べ，信任状が正しく書かれているかどうかをサーバプログラムに知らせる．異なる流儀に対しては異なる種類の情報を提供される．Unix 流では利用者の UID を与えるのに対し，Secure RPC ではネットワーク全体におけるユーザ名を与える．

この情報の解釈法および離れたところにいるユーザに対してどのサービスが提供されるべきかに関する方策は，RPC 機構の範囲外である．RPC は情報を離れたところに送り，その答を受け取るだけである(図 5-4 参照)．



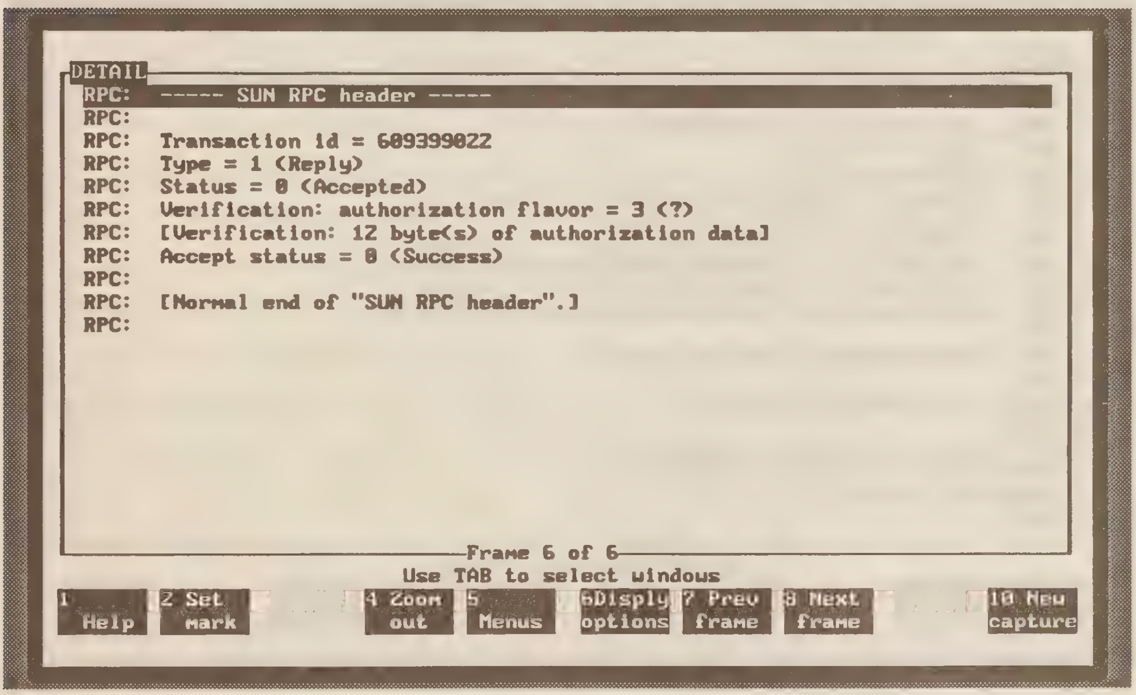
Courtesy of Network General

5-2　RPCヘッダ



Courtesy of Network General

5-3 認可の流儀の付加



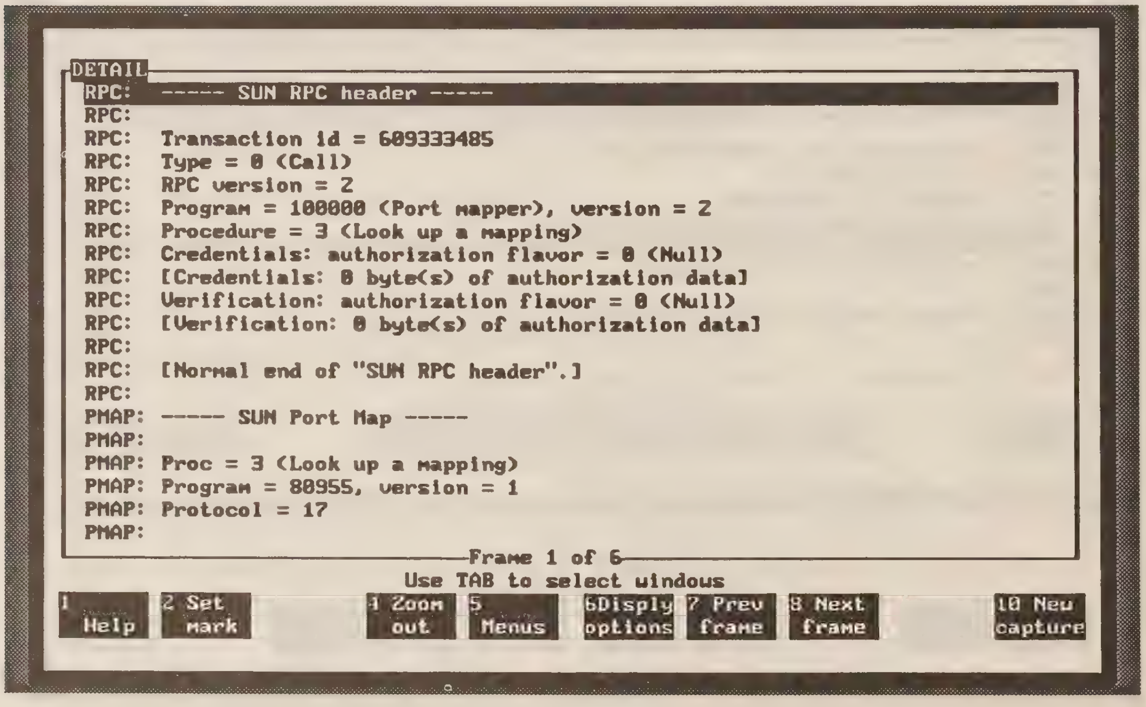
Courtesy of Network General

5-4 認可の受理

5.2 ポートマッパ

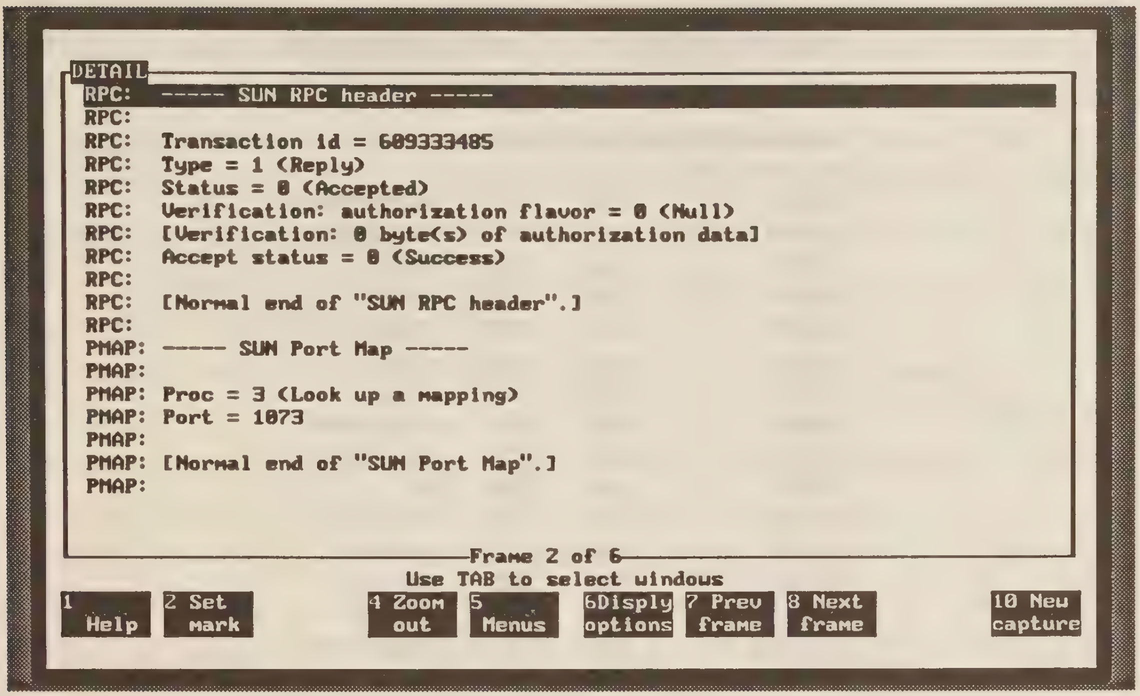
サーバ上のプログラムの現在のアドレスを見つけるために，ポートマッパまたは rpcbind サービスが用いられる．ポートマッパはプログラム番号 100 000 が割り当てられた RPC プログラムである(図 5-5 参照)．この RPC プログラムは周知の TCP や UDP のポートを listen する唯一のものである．RPC 呼出しを受け付けるように設定されたどんなシステムでも，少なくともこのアプリケーションだけは動いている．図 5-5 では，クライアントプログラムがポートマッパにプログラム番号 80 955 のプログラムの現在の状態を調べるための要求を送っている．図 5-6 では，プログラムが目下 1073 番ポートで呼出しを受けていることを示している．

この時点で，宛先のプログラムへ TCP コネクションを張る(もしくは UDP データグラムを送る)プログラムがほとんどである．われわれはこれまで，この特定のノード上にプログラムが存在することをクライアントプログラムが前もって知っていることを仮定してきた．サービスの現在の位置を見つけるための方法の 1 つは，NIS ネームサーバというもう 1 つのプログラムを使うことである．典型的なクライアントの初期化においては，サービスを提供するホストシステムの名前を要求する．そして，そのホストシステム上のポートマッパと連絡をとってサービスの現在のアドレスを見い出す．次いでクライアントは，RPC クライアントライブラリを通じて，ポートおよび



Courtesy of Network General

5-5 ポートマッパ



Courtesy of Network General

5-6 ポートマッパの応答

ネットワークアドレスの特定の組合せに対するコネクションを TLI が確立することを要求する.

よく知られた RPC サービスの多くはサーバシステム上でブート時に起動されるデーモンとして実現される. 中には, 重要なサービスを提供するが滅多に呼ばれないプログラムもある. このようなプログラムに対しては, コネクション要求の到来をトランスポートポートで listen し, ポートがメッセージを受け取ったときにそのサービスを呼び出す, Internet デーモン (inetd) と呼ばれる特別なデーモンを代理として用いることができる.

図 5-7 にはポートマッパを用いてどのプログラムが準備ができていて, 要求を待っているかを調べる rpcinfo のコマンドの例が 2 つ示されている. 1 つ目のコマンドは commserv というサーバから, NFS セッションを起動するために用いられる "mount" サービスの情報を求めている.

commserv からの応答は, サーバでは 2 種類のバージョンのマウントプログラムが利用可能であることを示している. rpcinfo コマンドはまず NIS に問い合わせてホスト "commserv" のネットワークアドレスを知る. そして, ポートマッパは特定のプログラムの状態を知るためにホスト "commserv" に要求を送る.

図 5-7 の 2 つ目のコマンドに示すように, あるシステム上で現在利用することのできるすべての RPC に基づくプログラムを見い出すことができる. これらのプログラムの多くについては後の章で述べる. 例えば, ypbind は NIS のネームサーバを探しているクライアントが使うプログラムである. keyserv は Secure RPC で公開鍵暗号のために使われる.

```
hydramatic% rpcinfo -u commserv mount
program 100005 version 1 ready and waiting
program 100005 version 2 ready and waiting

hydramatic% rpcinfo -p commserv
program vers  proto  port
100000  2     tcp    111    portmapper
100000  2     udp    111    portmapper
100007  2     tcp    1024   ypbind
100007  2     udp    1027   ypbind
100007  1     tcp    1024   ypbind
100007  1     udp    1027   ypbind
100029  1     udp    665    keyserv
100028  1     tcp    670    ypupdated
100028  1     udp    672    ypupdated
100003  2     udp    2049   nfs
100005  1     udp    717    mountd
100005  2     udp    717    mountd
100005  1     tcp    720    mountd
100005  2     tcp    720    mountd
100026  1     udp    725    bootparam
100024  1     udp    729    status
100024  1     tcp    731    status
100021  1     tcp    732    nlockmgr
100021  1     udp    1037   nlockmgr
100021  3     tcp    737    nlockmgr
100021  3     udp    1038   nlockmgr
100020  1     udp    1039   llockmgr
100020  1     tcp    742    llockmgr
100021  2     tcp    745    nlockmgr
100021  2     udp    1040   nlockmgr
100011  1     udp    1042   rquotad
100001  2     udp    1043   rstatd
100001  3     udp    1043   rstatd
100001  4     udp    1043   rstatd
100012  1     udp    1045   sprayd
100008  1     udp    1046   walld
100109  10    udp    1047   activity
100118  10    udp    1048   etherif
100117  10    udp    1050   hostif
100112  10    udp    1051   hostmem
```

5-7　利用可能なRPCサービス

5.2.1　トランスポート選択の問題

トランスポート機構の選択は通常はネットワークの低位層にまかされている．プログラマはまったく気にしないかもしれない．しかし，違いを生じ得る場合もある．

例えば，TCPは信頼できるプロトコルであるのに対して，UDPはそうではない．RPCはUDP環境ではそれに応じてより積極的な再試行を行うよう再送方策を変えるが，UDPでは遠隔サーバがまったく要求を受け取らないこともあり得る(ありそうなことではないが)．サーバが要求をまったく受け取らなければ，クライアントは確かには応答を受け取らない．したがってRPCコードはタイムアウトしてエラーメッセージを返す．しかし，プログラマはたかだか1回のセマンティクス(at-most-once semantics)(リクエストの配送が複数回ではなくて，1回だけもしくは1度もないこと)として知られることを保証したいかもしれない．TCPは，たかだか1回のセマンティクスを提供するトランスポートプロトコルの一例である．

TCPではなくUDPを使うことを正当化するためにしばしば持ち出される問題点の1つは，プロトコルのオーバーヘッドである．1つのパケットについては，UDPはTCPよりも間違いなくオーバーヘッドが小さい．しかし，注意深くコーディングされたTCP実装や複数のパケットが交換されるよくあるシナリオでは，TCPのオーバーヘッドはかなり小さくなる．

非常に長いデータを送らねばならないときにはTCPが有利になる傾向がある．UDPではデータグラムの大きさが8キロバイトに制限されている．そのため非常に長いデータを送らねばならないサービスではTCPが使われる．一方，TCPでエラーが生じると，ユーザプログラムはコネクションをリセットしなければならない．UDPでは，単に成功するまで(サーバがいなくなったり，クライアントがバインディングをリセットしなければ)試み続ければよい．ネットワークファイルシステム(NFS)はUDPを用いるRPCアプリケーションの一例である．

さらに，UDPは軽量だと考えられている．複数のクライアントからアクセスされるサービスでは，コネクションの管理はやっかいなものになりかねない．UDPは多くのクライアントをサポートするサーバに用いるのに簡単である．

トランスポート独立RPCでは，ユーザはトランスポートメカニズム選択の問題からさらに遠ざかることになる．典型的には，伝達機構は3つのクラスの1つに入る：コネクションレス型(データグラム)，不意の解放のあるコネクション型(例えばTCP)，手順を踏んだ解放をするコネクション型の3つである．プログラムはこれらのうちどれを望むかを指定したり，決定をユーザやデフォルトに委ねることもできる．/etc/netconfigというファイルの中に，利用できるトランスポートのリストが作られる．各々のユーザやプログラムは，$NETPATHという環境変数でどれを希望するか指定することができる．

TI-RPCの世界では，アドレス解決をどのように行うかが興味深い問題である．結局，トランスポートが決定されなければアドレスを知ることなどできない．ポートマッパの後継者であるrpcbindがこの問題を扱う．サービスは自分自身をrpcbindに登録する．

5.2.2 ブロードキャストRPC

ブロードキャストRPCによって，1つの呼出しをサブネットワーク上の複数のサーバに送るこ

とができる．ブロードキャスト RPC は，データ収集のようなアプリケーションや，最も近い NIS ネームサーバのような位置に依存しないサーバを見つけることなどに用いられる．もしすべてのサーバに同時に要求がばらまかれるなら，最初に返事をしたサーバが利用可能なリソースを最も多くもっているということ――負荷平均化の素朴な形態――を仮定することができる．

ブロードキャスト RPC 呼出しを作成する際の基本的な問題は，応答をどうやって待つか，また応答をいくつ待つかということである．最も早く応答するサーバにだけ興味があるなら，1 つの応答だけでよい．

ブロードキャスト RPC 機構を使うのに，サーバアプリケーションを変更する必要はない．なぜなら，クライアントが複数のサーバに働きかけているということは，あるサーバにとって何も関係のないことだからである．クライアントはどのようにして複数のセッションを維持するかを解決しなければならない．もちろん，サーバアプリケーションの中に複数のサーバの存在を知らせるセマンティクスを作ることもできるが，プロトコルにはそれを必要としない．

ブロードキャストは確かにトランスポート機構が問題となる例である．例えば，TCP はブロードキャストサービスを提供しないので，UDP が代わりに使われねばならない．UDP ではデータグラムの大きさに制限があるので，ブロードキャストパケットが 1 つの UDP パケットに収まるだけでなく，下位のネットワークの最大転送単位(MTU)にも収まらなければならない．イーサネットでは，MTU は 1500 バイト程度である．

もし下位のネットワークのプロトコルが IP ならば，さらにもう 1 つの問題に言及する必要がある．IP プロトコルでは，ブロードキャストは使用できるが，ネットワーク上のそれぞれのマシンの同じ IP ポートアドレスに行かねばならない．この基準に合致するサービスは，ポートマッパだけである．それゆえ，すべてのブロードキャストはポートマッパに振り向けられ，ポートマッパがマシン上のしかるべきサーバに転送する責任を担っている．もし適切なサーバが存在しなければ，ポートマッパはブロードキャスト要求を無視する．

ブロードキャスト RPC の一例は，引数をもたない rusers コマンドである．rusers によってマシンのユーザが誰であるかがわかる．もし rusers コマンドが引数をもたなければ，ブロードキャストが送られる．

RPC ブロードキャストを含むブロードキャストは，どんな種類のものでも広範囲に広がったときには，あらゆる種類の輻輳の問題を簡単に起こし得る．このため，IP ブロードキャストはゲートウェイを越えて転送されることはなく，1 つのローカルネットワークの中に(あるいはさらにサブネットワークの中だけに)制限される．

5.3 非同期機構

通常，RPC の機構は同期的である．すなわち，要求が送られると，その要求を送ったプロセス

は応答が返ってくるまでブロックされる．この基本的な動作を変えるものとして 2 つの方法が使われる．

・ノンブロッキング RPC
・コールバック RPC

ノンブロッキングは一方通行のメッセージ機能である．つまり，応答のためのパスは用意されない．典型的には，一連のノンブロッキング呼出しが送られた後で，サーバからの反応を得るために通常のブロック型 RPC 呼出し，あるいはコールバック RPC 呼出しが送られる．ノンブロッキング要求というのは，単にタイムアウトが 0 に設定され，タイムアウトエラーが無視される要求のことである．クライアントコードは結果を受け取らず，サーバの手続きは要求に応えないだけである．

ノンブロッキング要求が用いられるときには，RPC ライブラリはもはや動作がうまくいったかどうかを保証できないことに注意しよう．UDP によって送られた要求はどこかで失われるかもしれないが，送り側ではそのことを知ることができない．TCP が使われていれば，たかだか 1 回のセマンティクスが保証される．

コールバック RPC はノンブロッキングリクエストや，要求と応答からなる普通の同期呼出しに関して用いることができる．コールバック RPC は，一時的なアドレス空間を用いてクライアントとサーバの役割を入れ換えることにより，等位パラダイムに近いものを実現している．ひとつのクライアントが複数のサーバからのコールバックを受け取ることができる．

コールバック RPC 機構の典型的な例はメールシステムであろう．メールシステムが新たなメッセージを受け取ると，ユーザエージェントと連絡をとる必要がある．通常は，ユーザエージェントがクライアントであり，メールシステムがサーバである．

この仮想的なアプリケーションにおけるコールバック RPC においては，サーバは最初クライアントとして始まる．つまり，ユーザエージェントを呼び出してメールが届いたことを知らせる．少したってから，ユーザエージェントは与えられたアドレスを用いてクライアントとして呼出しを行い，メールを受け取る．

ここで使っている方法は，クライアントが一時的なプログラム番号を用いて擬似的なサービスを作り出すことである．プログラム番号は一時用の範囲からランダムに選ばれ，番号が重ならないことを保証するためにマシン内のポートマッパに登録される．そうして，その情報が RPC 要求の一部としてサーバに送られる．

この時点で，サーバは典型的には当初の要求に対して単なる確認しか返さないことに注意する必要がある．サーバが何らかの仕事をするにはある長さの時間がかかる．要求を処理すると，サーバは与えられた一時的な番号に対してコールバックを起動するだろう．途中の確認は，最初の要求をタイムアウト 0 のノンブロッキングにすることでなくすことができる．

コールバックを扱うために，クライアントは自分自身をサーバとして設定する必要がある．ser-

vice run システムコールの呼出しを通じてこれを行う．

5.3.1 バッチモード RPC

バッチモード RPC は，応答を受け取る前に一連の呼出しすべてを一度に送るための方法である．一連のノンブロッキング RPC コールが送られる．そして，タイムアウトが 0 でない要求が送られ，サーバへの現存するコールがすべて一掃される．おそらく，サーバアプリケーションは最後の呼出しを受け取るまで応答を返さないようになっているだろう．パケットが失われないという点で，トランスポート層としては TCP が推奨される．

5.4 XDR ライブラリパッケージ

XDR ライブラリパッケージには，C 言語の基本的なデータ構造を XDR 形式の構造のシリアルストリームに変換したり，その逆を行うルーチンの集まりが含まれている(図 5-8 参照)．同じルーチンがどちら向きの変換も(さらに文字列のような可変長構造へのメモリアロケーションも)行う．

それぞれ C ルーチンが一群の基本型への変換を扱う．さらに，これらを組み合わせることで，リンクされたリストや構造体のようなものも扱えるようになる．ポインタについてもサポートされている(ポインタには特別のサポートが必要である．というのは，ポインタの値はマシンアドレスであり，他のマシンではまったく意味をもたないからである)．

XDR の基本データタイプ(図 5-9 参照)を組み合わせて様々な複雑なデータタイプを作ることができる．基本データタイプはすべて 32 ビット単位で表現されることに注意されたい．すなわち，8 ビットの文字の場合 32 ビット全部が使われ，24 ビットは 0 である．文字列については，データはまとめられ，次の 32 ビットの境界に達するまで詰めものがされる．

ルーチンが使う実際のデータは XDR ストリームからくる．XDR ストリームは何種類かの異なる源からくることができる．

- メモリ
- レコードマーカのある TCP/IP
- Unix の標準 I/O(ファイルやデバイスを含む)

TCP は XDR ライブラリに対してストリームに似たインターフェイスを提供するが，それによって生じる問題がある．つまり，ある構造体がいつ終わり別の構造体が始まるのかをライブラリにはわからない．XDR は，レコード境界を挿入し，これが相手側の XDR で用いられる．XDR はまた Unix の標準 I/O もサポートしているので，ファイルだけでなく，基本的な read および write 呼出しをサポートすることのできる他のどんなデバイスもサポートできる．

XDR はネットワークに出て行こうとするすべてのデータを受け取って，XDR 標準フォーマッ

ルーチン	解説
データ型プリミティブ	
xdr_array()	配列の変換
xdr_bool()	論理型の変換
xdr_bytes()	長さを指定された文字バイト列の変換
xdr_char()	文字型の変換
xdr_double()	倍精度実数型の変換
xdr_enum()	列挙型の変換
xdr_float()	浮動小数点型の変換
xdr_int()	整数型の変換
xdr_long()	long int型の変換
xdr_opaque()	固定長の不明瞭なデータの変換
xdr_reference()	構造体の中のポインタの追跡
xdr_short()	short int型の変換
xdr_string()	nullで終わる文字列の変換
xdr_u_char()	符号なしキャラクター型の変換
xdr_u_int()	符号なし整数型の変換
xdr_u_long()	符号なしlong int型の変換
xdr_u_short()	符号なしshort int型の変換
xdr_union()	識別された共用体の変換
xdr_void()	常に真を返す
xdr_wrapstring()	RPCルーチンからのコールに対し文字列をひとまとめにする
ストリームの生成	
xdrmem_create()	XDRメモリストリームの初期化
xdrrec_create()	TCPトランスポートのレコード境界のあるXDRストリームの初期化
xdrstio_create()	XDR標準I/Oファイルストリームの初期化
xdr_destroy()	ストリームの破棄と関連するメモリーの解放
その他	
xdr_inline()	XDRストリームに関連するインラインルーチンを起動する
xdr_getpos()	XDRストリーム上での現在の位置を返す
xdr_setpos()	XDRストリーム上での現在の位置を変える
xdrrec_endofrecord()	XDRレコードストリームにレコード終端子でマークする
xdrrec_eof()	XDRレコードストリームにファイル終端子でマークする
xdrrec_skiprecord()	XDRレコードストリーム内の残りのレコードをスキップする
出典：Sun Microsystems	

5-8 XDRライブラリコール

符号付き整数	8,16,32,64ビット
符号なし整数	16,32,64ビット
IEEE浮動小数	32,64ビット
論理型	
数え上げ型	
固定長 あるいは 可変長 配列	構造体の配列を含む，任意の型の配列
固定長 あるいは 可変長 バイト文字列	
構造体	
識別された共用体	
隠蔽されたデータ(Cookies)	

5-9 XDR基本データ型

トに変換する．XDR はデータの標準(正規)ネットワーク表現を定義している．すべてのクライアントおよびサーバは，この標準形式との間でデータを変換する．Sun アーキテクチャを採用した 2 つのコンピュータ(つまり，Motorola 68000 と SPARC)の場合には，XDR のアーキテクチャは内部表現に近いものになっている．もちろん，これは XDR がする仕事が少なくて済むということを意味する．

変換の最悪のケースは，VAX が別の VAX に通信する場合である．VAX は Sun とは内部表現が大きく異なっているので，データを XDR フォーマットに変換するのに若干の努力を要する．相手側の VAX で，もう 1 度データが変換される．

5.4.1 XDR の利用

XDR を RPC と共に使うこともできるし，単独で用いることもできる．アプリケーションは情報の符号化を定義する．例えば，データを単に長いバイト列として符号化することもできる．離れたところにあるファイルを NFS がクライアントプログラムに対して提供する際には，この方法を使っている．

これは離れたところにあるファイルを自計算機上のファイルと同様に見せようとする NFS のようなサービスの場合には適当である．NFS はデータの中身については何も行わない．データが何を意味しているかを解釈するのはプログラムの責任である．あるファイルはディスクレスノードのためのオペレーティングシステムであり，あるものはオペレーティングシステムによって実行されるワードプロセッサで，またあるものは PostScript を理解するデバイスによって解釈されるグラフィックファイルかも知れない．

グラフィックファイルや，その他のデータファイルは，XDR を用いて情報にさらに構造を提供することのできる一例である．いくつかのファイルからなる中央データソースの例を考えよう．

ネットワークのあちこちで一連のプログラムが異なるマシンアーキテクチャの上で動作しているが，どのプログラムも同じデータにアクセスしている．(例えば，VAX上の)あるアプリケーションプログラムがデータをファイルに書き込み，また別の(例えばSun上の)アプリケーションプログラムがそのデータファイルを読み出そうとするかもしれない．NFSは，RPCやネットワークについての知識をもつことなしにデータを提供する．ファイルはSunとVAXのどちらからでも，自分のところにあるかのように見える．VAXは整数をある形で書き込むが，Sunはその整数を別の形で読み出すだろう．

5.4.2 XDRの使用例

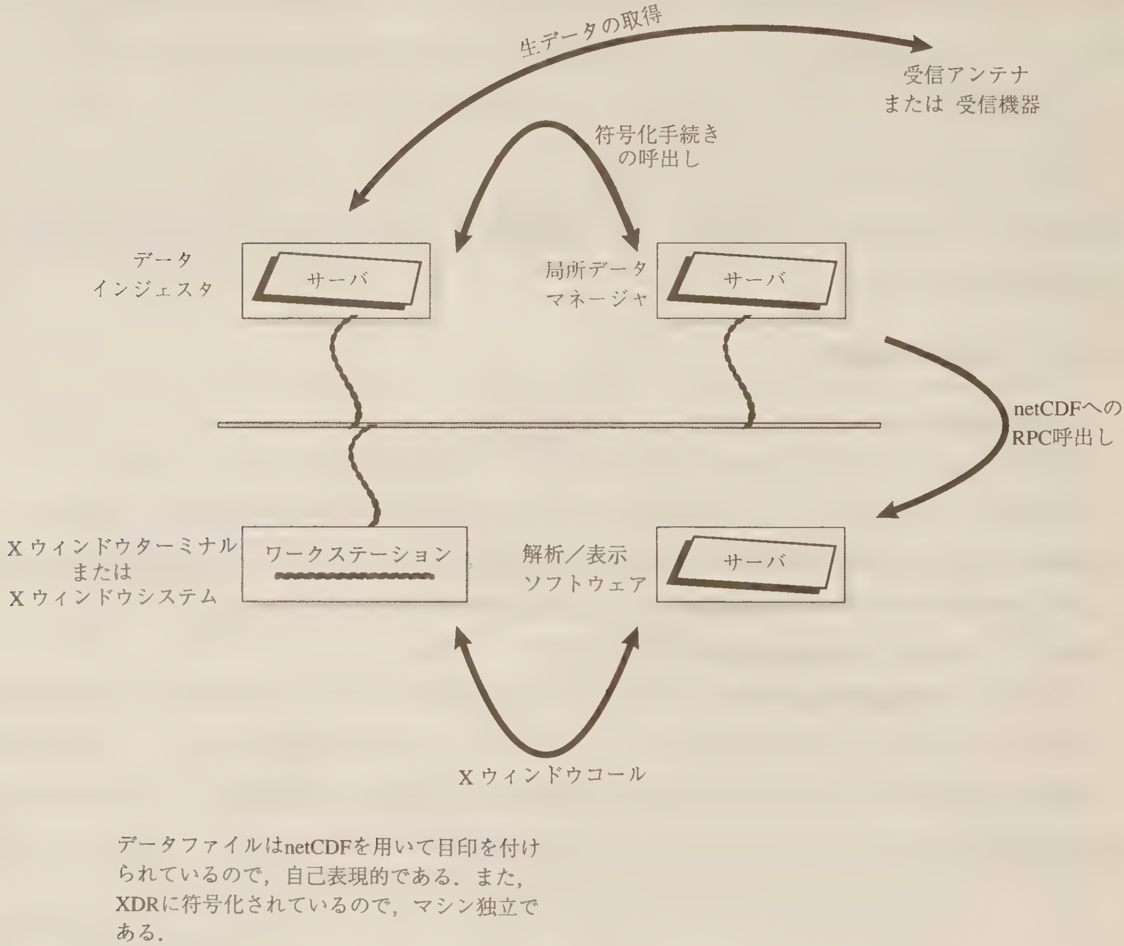
この問題の簡単な解決策は，直接ファイルに読み書きするのでなく，SunとVAXの両方がファイルへのXDRストリームを用意することである．この方法では，データはマシン独立なフォーマットで格納される．これは，UnidataがUniversity Corporation for Atomospheric Researchのために書いたプログラムで使われている．Unidataは気象データを大学やその他の機関に安価に配布するためのアーキテクチャを開発することを委託された．図5-10にあるように，基本的なアーキテクチャはRPCとXDRを基礎にしている．

人工衛星や受信装置を通じてデータがキャンパスまでやってくると，データインジェスタ(データを摂取するもの)として知られるコンピュータによってデータが受け取られる．データインジェスタはこのリアルタイムのデータストリームを受け取り，netCDFに変換してファイルサーバに蓄える．netCDFはXDRの上にさらに上位の層を築くものである．XDRはデータをマシン独立なフォーマットで表現する方法を提供するが，離れたノードに正確な構造を知らせる方法はもっていない．たいていのRPCアプリケーションでは，入ってくるデータの構造に関するこの知識は，XDRのパック，アンパック手続きに作り込まれる．このことは，この特定のデータストリームがすでに定義され，それがXDRルーチンの中に作られていることを仮定している．

CDFとはNASAのCommon Data Formatのことで，データが自身を記述する方法を提供する．netCDFはデータが何であるかを記述するのにNASAの方法を採用し，それを(いくつかの拡張を含んだ)XDRのデータ表現法と組み合わせている．データはこのnetCDFフォーマットで格納されるので，ひとたびコンピュータ内のデータマネージャに落ちつけば，自己表現的である．この時点で，いろいろな解析プログラムがデータを呼び出すことができる．netCDFライブラリによって，非常に大量な2次元あるいは3次元のデータに簡単にアクセスすることができる．

この情報を利用できるいろいろなアプリケーションプログラムがある．このような解析あるいは表示のためのソフトウェアは，様々な源から得られている．例えば，図5-11はPurdue's University Weather Process(WXP)というソフトウェアの画面の一例である．このソフトウェアは情報を集めるためにコンピュータ内のデータマネージャを呼び出している．

データを実際に表示するのは，X-ウィンドウターミナルやX-ウィンドウのサーバプロセスが



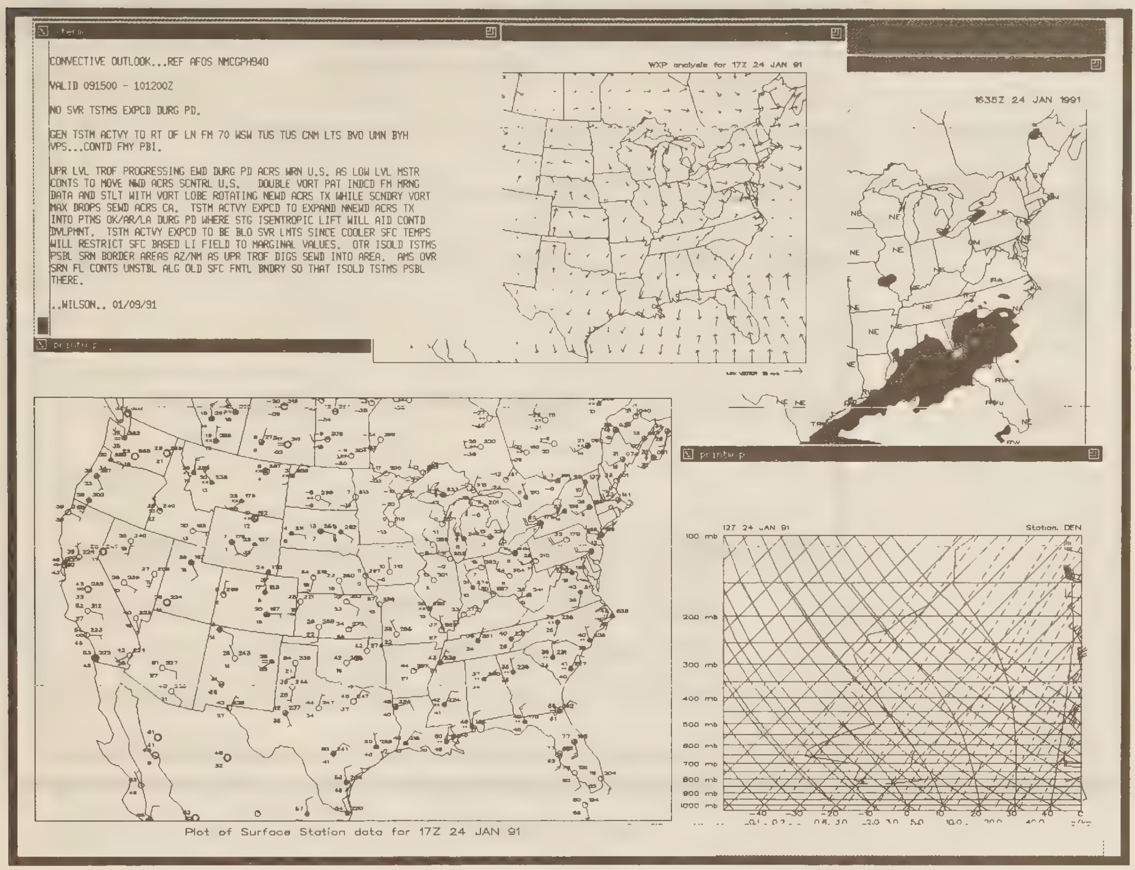
5-10　netCDFデータ収集システム

動いているその他のワークステーションのような，別のコンピュータ上でも十分に可能である．表示ソフトウェアはデータを解析する目的に関してはサーバであるが，画面にデータを描く目的に関してはワークステーションのX-ウィンドウサーバのクライアントとなる．

RPCプロトコルは，クライアントがネットワークを横断して遠隔手続き呼出し要求を出せるようにするための基礎を提供する．そのアプリケーションプログラムは，情報を画面に表示するのにウィンドウシステムのような他のサービスを利用することもできる．

5.4.3　RPCの例

図5-12は，今度はSun Microsystemsによって書かれたRPCアプリケーションのプログラムの例を示している．多くの会社同様，Sunも従業員に株式購入オプションを与えるのが好きである

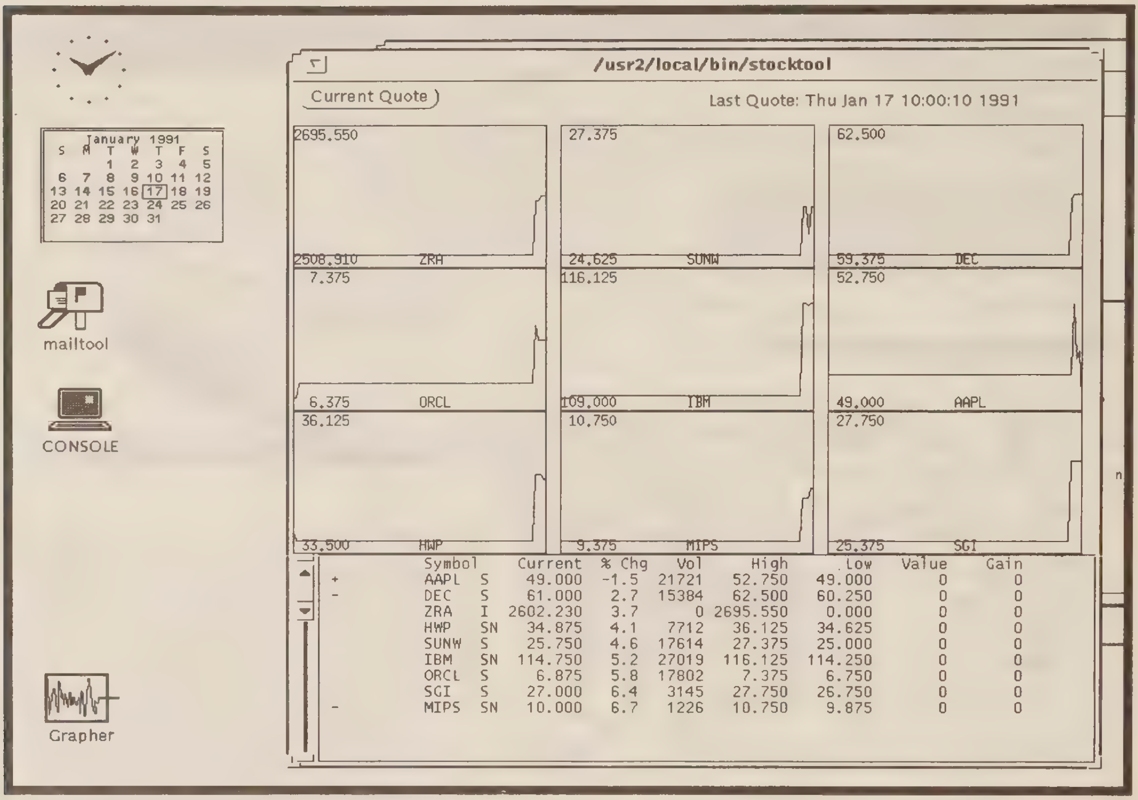


5-11　気象ツール

Courtesy of Unidata

が，これは大企業で従業員の労働意欲を増すための共通した方法である．従業員がラジオのビジネスレポート番組を聞くのではなく，本来の仕事に集中できるよう，利口なプログラマが図 5-12 にあるような RPC Stock Tool を開発した．このツールは，いろいろな株式の価格の相場を得るために RPC プログラムを用いている．この例では，Sun の株価が他のコンピュータ各社と比較されていることに気づくであろう．株式ツールプログラムはデータを得るために RPC を使っているが，そのデータを表示するためにウィンドウシステムを呼び出している．Sun では，ユーザがウィンドウシステムをあるサーバからマウントし，アプリケーションプログラムを別のサーバからマウントするのは珍しいことではない．

図 5-12 には，他にもいくつかの RPC に基づくプログラムが示されている．ユーザが自分のカレンダーを他の人と調整することのできるカレンダーツールに気づくであろう．このカレンダーには予定を思い出させる機能もある．カレンダーの下にあるのはメールツールである．メールボックス上の小さな旗が立っていることに注目されたい．これは新たにメールが到着したことを示している．メールツールの下の方あるのは，“grapher”のアイコンであり，これは SunNet Manager



Courtesy of Sun Microsystems

5-12 Sun株式ツール

がネットワーク管理情報を画面に表示するために用いるものである．SunNet Manager は，ネットワーク中で他の管理対象ノードの管理情報を得るために RPC を用いる．

より進んだ勉強のために

John Corbin, *The Art of Distributed Applications—Programming Techniques for Remote Procedure Calls.* Springer Verlag (New York: 1991).

Sun の上級研究員による，ONC RPC のメカニズムに限定的に注目したもの．

Sun Microsystems, *Network Programming Guide,* Part Number: 800-3850-10, Revision A of 27 March, 1990.

Revision A of 27 March, 1990. RPC, XDR の仕様とプログラミングへの入門的な資料とプロトコル仕様，および RPC と rpcgen のチュートリアルを含んでいる．

———, *RPC: Remote Procedure Call Protocol* (ver. 2). RFC 1057.

———, *XDR: External Data Representation Standard.* RFC 1014.

Chapter 6

ネットワークファイルシステム

6.1 NFS

ファイルシステムがローカルディスクにマウントされる場合，マウントされるのはファイルシステム全体である．ディスクはパーティションに分かれており，それぞれのパーティションの中身が1つのファイルシステムとなる．NFSを使うとファイルシステムをネットワークを介したものに拡張することができる．しかし，違いもある．パーティション全体ではなく，リモートファイルシステムのパーティションの一部だけでもマウントすることができる，という点である．これによってユーザは，実行ファイルやユーザスペースやその他のデータを含んだファイルシステム全体の代わりにマニュアルページだけをとる，というようなこともできる．

サーバ側ではNFSは3つの要素からなっている．あるファイルの集まりをネットワークのユーザに対して利用可能とすることを指定するにはexportfsコマンドを使う．このコマンドではいくつかのオプションを指定でき，ファイルシステムを読出しのみに限定したり，アクセスを特定のユーザに限定したり，さらには他のホストからのrootアクセスを許すかどうかなどの指定ができるようになっている．

残りの2つの要素はサーバ上のデーモンである．rpc.mountdは特にブートストラップ問題だけを扱うデーモンである．ファイルの使用をクライアントからあらかじめサーバに通知させるようにすることによって，サーバはクライアントを認証することができるようになっている．また，このデーモンによってサーバはファイルハンドルを返すことができる．ファイルハンドルというのは，どのファイルが参照されているのかをサーバ側が判別できるように，それ以後の要求の際にクライアント側が提示するデータのことである．

要求のたびにファイルハンドルをやりとりすればいいと思うだろう．実際そうなっている．しかし最初の参照を扱うために，システムとは独立の手段が必要である．サーバはファイルハンドルの中にその対象の場所を知るための情報を含めている．ファイルハンドルとコマンド(ディレクトリ中のオブジェクトを読むなどといったもの)が示されると，ファイルシステムは名前のリスト

を返す．lookupコールを使って名前からそれぞれのファイルハンドルを求めていく．マウントプロトコルはファイルハンドルのブートストラップなのである．つまり，違うファイル名の規則をもつ異種のファイルシステムがすべてファイルシステムを公開できるようにするのである．

3つ目はNFSデーモン自体である．NFSデーモンは送られてくるNFSコールを処理して，それに対する答えを送り返す．デーモンは異なる複数のユーザを相手にすることができる．プロトコルは内部状態をもたないもの(stateless)であり，要求の間で何の情報も保持されないし接続も存在しないということを思い出してほしい．サーバでは8個のデーモンが起動されているのが普通である．つまり，同時に8個までのNFS操作を処理することができる．

6.1.1 ファイルシステムの公開

exportfsコマンドはネットワーク上のクライアントがファイルシステムを利用できるようにするために使われる．スーパーユーザが次のようにexportfsコマンドを使って手動で公開してもよい．

```
/usr/etc/exportfs -a
```

このコマンドは/etc/exportsファイルのすべての行を読み込んで情報をカーネルに送る．特定のディレクトリだけを公開(あるいは公開取消)することもできるし，exportsファイルには書かれていないオプションを加えることもできる．たいていのサーバでは，このコマンドはシステムがブートするときに1回実行されるようになっている．

コマンドには様々なオプションがあり，ファイルシステムがどのように公開されるかを制限(あるいは拡大)することができる．例えば，マニュアルページのように変更の必要がない共有ファイルシステムは読出し専用(read-only)で公開されることが多い．

その他のrootやaccessといったオプションは，情報の利用をあるユーザのグループに限定する

ro	読出し専用として公開する．(デフォルトは読み書き可能)
rw=ホスト名	大半が読出しとして公開する．つまり，大半のホストに対しては読出し専用として，指定されたホストに対しては読み書き可能として公開する．
anon=ユーザID	未知のユーザからの要求に対しては実効ユーザIDとして指定されたユーザIDを用いる．ルートユーザ(ユーザID=0)は以下に示すrootオプションに含まれていない限りは未知のユーザとサーバはみなす．デフォルトは-2(nobody)である．-1に設定することによって匿名アクセスを不可能にすることができる．
root=ホスト名	指定されたマシンのルートユーザからのルートアクセスだけを許す．
access=クライアント	(ホスト名またはネットグループで)リストアップされたクライアントにマウントアクセスを許す．デフォルト値ではすべてのマシンがディレクトリのマウントを許される．
secure	secure RPCをアクセスのときに要求する．

6-1 NFS公開オプション

必要がある場合に用いられる．ルートアクセスは通常，ディスクレスクライアントがサーバのルートファイルシステムにルート権限でアクセスできるようにするために用いられる．これに関連して重要なのは anonymous 公開オプションで，これは登録されていないユーザがアクセスを要求してきた場合にどうするかを設定するものである．

公開されたファイルシステムへのアクセス制御はクライアント単位で行われるように実装されており，公開パラメータによって十分にきめ細かな制御を行うことができる．ユーザ識別子が与えられれば，exports ファイルからどのファイルシステムとどのようなオプションが許されているのかが決定できる．ユーザの安全な認証(secure authentication)は，アクセス制御とはまた別の話であり，secure NFS サービスによって扱われる．

6.1.2 ファイルシステムのマウント

公開(export)はファイルシステムをネットワークに対して利用可能にする手段である．ファイルシステムを実際にマウントするには，クライアントはマウントコマンドを発行しなければならない．このマウントコマンドを発行するには 3 種類のやり方がある．

- ファイルシステムテーブルからのワークステーション初期化の一部として行う．
- ルート権限をもつユーザが対話的に行う．
- オートマウンタを使って行う．

オートマウンタについては 11 章でより詳しく解説する．このサービスは動的なマウンタであり，大きなネットワークではこれを使うことにより NFS の管理がかなり簡単になる．通常，重要なファイルシステムはブート時にマウントされる．例えばディスクレスクライアントは，Unix コマンドを実行できるようにオペレーティングシステム，つまりルートとユーザファイルシステムをマウントする必要がある．おそらくワークステーションはさらにユーザのホームディレクトリやマニュアルページ，必要な共通ソフトウェアシステム，その他ワークステーションが必要とするデータのあるファイルシステムもマウントする必要があるだろう．

図 6-2 はファイルシステムテーブルである/etc/fstab に含めることのできる情報を示したものである．ただし，この他にも様々なファイルシステムをマウントすることができるので，この表にはさらに追加があることに注意してほしい．NFS についてみると，マウント操作の進行を制御する様々なオプションがある．

マウントパラメータとしては，ローカルの NFS クライアント動作を制御するもの，例えばどの程度再試行に力を入れるかといったものが多い．この制御の程度により，パラメータを調整して性能を向上させる(あるいは悪くすると再試行過剰でネットワークの迷惑な住人になってしまう)ことができる．

hard と soft マウントオプションはクライアントにとって非常に重要である．soft マウントとは，サーバが応答しなくなったクライアントは接続をあきらめて，アプリケーションプログラム

ファイルシステムタイプ	
4.2	4.2bsdブロック特殊デバイス
nfs	公開されたNFSファイルシステム
swap	スワップパーティション
rfs	RFSファイルシステム
tmp	仮想記憶上のファイルシステム
hsfs	High Sierra 標準CD-ROM
pcfs	DOSファイルシステム(ディスケット)
NFSオプション	
bg | fg	最初に失敗したら，バックグラウンドで(bg)またはフォアグランドで(fg)再試行を続ける.
noquota	ユーザがそのファイルシステムでクォータを越しているかどうかはチェックしない.
retry=n	マウント操作を再試行する回数
rsize=n	読出し転送バッファサイズ
wsize=n	書込み転送バッファサイズ
timeo=n	NFS再送のタイムアウトを十分のn秒にする.
retrans=n	1回のRPC要求でのNFS再送の最大回数
port=n	サーバのIPポート番号
soft/hard	サーバが応答しないときにエラーを返すか，サーバが応答するまで要求を繰り返すか
intr	ハードマウントでキーボードからの割込みを受け付ける
secure	secure RPC認証を用いる
agregmin=n	ファイルが変更されてからn秒の間属性のキャッシュを保持するレギュラーファイルでの最小値
agregmax=n	レギュラーファイルでの最大値
acdirmin=n	ディレクトリでの最小値
acdirmax=n	ディレクトリでの最大値
actimeo=n	レギュラーファイルとディレクトリの最小/最大時間をn秒に設定する
noac	属性キャッシュを抑制する
デフォルト	fg,retry=10000, timeo=7, retrans=3, port=NFS_PORT, hard,agregmin=3, agregmax=60, acdirmin=30, acdirmax=60

6-2　Fstab NFSオプション

にはエラーを返す，という意味である．hard マウントの場合は，クライアントは接続を試み続ける．ホームディレクトリの場合などファイルシステムが重要な書込み動作をする場合には hard マウントが必要である．

hard マウントは，対象のサーバが単に一時的に過負荷になっているというのではなく，本当に停止してしまったときには問題を生じ得る．hard マウントを指定する場合には，割込み (intr) オプションも指定して，際限なく再試行を繰り返し続けるのをキーボードから止められるようにしておくのがよいことが多い．

図 6-3 は短い/etc/fstab ファイルの例である．最初の 2 つのマウントはローカルのディスクドライブであることに注意してほしい．あとの 3 つのマウントはそれぞれ違ったサーバからのもので

# /etc/fstab					
/dev/sd0a	/	4.2	rw	1	1
/dev/sd0g	/usr	4.2	rw	1	2
node1:/usr/man	/usr/man	nfs	bg,ro,soft,actimeo=3600	0	0
node2:/usr/news	/usr/news	nfs	bg,ro,hard,intr,secure	0	0
node3:/usr/local	/usr/local	nfs	bg,ro,soft,actimeo=3600	0	0

6-3 /etc/fstabファイルの例

ある．マニュアルページはserver1からで，このマウントはbackgroundマウントである．つまり，何らかの理由でマウントがすぐにできなかった場合，再試行は続けるが，その間ユーザが作業を続けられるようにするということである．

他の2つのマウントはUsenetのニュースファイルと，localファイルスペースのためのものである．localファイルスペースは読出し専用マウントであることに注目してほしい．属性値であるtimeoutは，クライアントがファイルに変更があったかどうかをチェックする頻度を表すものであるが，1時間に設定されている．これは，ファイルがそれほど頻繁に変更されないだろうと考えられているからである．

showmountコマンドを使うと，現在どのファイルシステムがクライアントにマウントされているかを見ることができる．またサーバにどのファイルシステムが公開されているのかを問い合わせることもできる．現在クライアントからリモートマウントされているディレクトリ階層がすべて表示される(ただし，クライアントがクラッシュしてしまい，サーバの情報が若干実際と合わない場合もあり得る)．

図6-4はshowmountコマンドの実行結果である．最初は，“commserv”という名前のサーバが公開しているファイルシステムを問い合わせたものである．公開されているファイルシステムそれぞれに，マウント操作を実行することのできる特定のグループが設定されている．

その次のshowmountコマンドは-aオプションを使って，サーバ上で動作中のすべてのマウントを表示している．図6-4のリストは，Sunのエンジニアリンググループにあるファイルサーバであるこのホストで動作中のマウントのごく一部である．

サーバが広範囲のクライアントに対して公開されるのと同様に，クライアントも多数の異なったサーバからファイルシステムをマウントする．対話的にファイルシステムをマウントするときに使われるのと同じコマンドを使ってワークステーションの現在の状態を見ることができる(図6-5)．hydramaticというワークステーションには，2つのサーバがファイルを提供していることに注目してほしい．

1つのサーバが多数の異なったワークステーションを扱うこともできるが，まったく同様に，いくつものサーバからのマウントによって非常に大きなファイルシステムを構成することもできる．2章でNASA-Ames研究センターにある空力数値シミュレーションセンター(Center for Numerical Aerodynamic Simulation : NAS)のネットワーク構成について説明した．このネットワー

```
hydramatic% showmount -e commserv
export list for commserv:
/usr                                    engineering,spd-topdomain
/export/home                            engineering,spd-topdomain
/var/spool/mail                         engineering,spd-topdomain
/export/exec/sun3.sunos.4.1BETA         engineering,spd-topdomain
/export/exec/kvm/sun3.sunos.4.1BETA     engineering,spd-topdomain
/export/root/majestic                   majestic
/export/root/osento                     osento
/export/swap/osento                     osento

hydramatic% showmount -a commserv
alexander:/export/home
alty:/export/home
anacapa.Corp.Sun.COM:/export/home
ark:/export/home
atticus:/export/home
baraka:/export/home
bari:/export/home
barry:/export/home
batcomfs:/export/home/dcw
bbeachco:/export/home
bigboss.Corp.Sun.COM:/export/home
billthecat:/export/home
bisun:/export/home
blizzard:/export/home
blond:/export/home
blueox:/home/commserv
bunny2:/export/home
bzone.Corp.Sun.COM:/export/home
calsun:/usr/share/man
centauri.Corp.Sun.COM:/export/home
chance:/export/home
chao:/export/home
charm:/export/home
cheers2:/export/home
chest:/export/home
chispa:/export/home
clown:/export/home
colorwriter:/home/commserv
comedy:/export/home
```

6-4　showmountのデータの例

クには2台の大型の Cray コンピュータシステムがあり, Ultranet のギガビットネットワークによって相互接続している．この Cray コンピュータシステムはどちらも Unix に加えて NFS プロトコルを動かしている．それぞれのシステムのファイルシステムは互いに公開されている．

```
hydramatic% mount -p
/dev/sd6a                          /                        4.2  rw  1 1
/dev/sd6g                          /usr                     4.2  rw  1 2
comm:/export/home                  /usr/export/home/ouch    nfs  rw  0 0
comm:/export/snm                   /ouch/snm                nfs  rw  0 0
swserver44:/export/local/sun3      /usr/local               nfs  ro  0 0
swserver44:/export/local/share     /usr/local/share         nfs  ro  0 0
```

6-5 ローカルにマウントされたシステム

図6-6は一方のCRAY YMPでdfとmountコマンドを実行した様子である．非常にたくさんのファイルがNavierというCRAY 2システムのホストからマウントされていることがわかる．CRAY 2の方で同じコマンドを実行しても，やはり同じようにホストReynolds(YMP)から多くのマウントが行われていることが表示されるだろう．例えば/navier/src3ファイルシステムは4キロバイトブロック170万個から構成されていて，マウントされたファイルシステムは全体で6.8ギガバイトの大きさになっている．

ここで重要なのは，クライアントはネットワーク中の1つあるいはそれ以上のサーバにデータを保存できるということである．ローカルファイルシステムにとっては，ファイルはローカルにマウントされているように見える．2つのシステムがどちらもUnixをベースにしたものであれば，マッピングはまったく単純である．

このサービスはその他のオペレーティングシステムでも動作する．例えばIBM/PCでは，ファイルシステムはPC-NFSを使ってマウントすることができ，ディスクドライブが追加されたように見える．IBM/PCの場合はこのような形にするのが適当であろう．VMSシステムの場合でもNFSマウントをすることができ，これもまた新しいデバイスとして見えるようになっている．

6.2 NFSプロトコル

ひとたびマウントサービスがファイルハンドルをクライアントに与えるという仕事を行ってしまうと，それ以降はNFSサービスが引き受ける．NFSプロトコルは内部状態のない(stateless)プロトコルである．つまり，すべての要求は他の要求とは独立性がある．

これは，何よりもクライアントとサーバの間でセッションが維持されることはない，ということを意味する．普通はUDPプロトコルが使われるので，トランスポート層でいうところのコネクション(connection)すら存在しない．NFSをTCPプロトコル上で動作させることも可能ではあるが，基本的なLANの実装はコネクションなし(connectionless)のものである．

内部状態のないプロトコルを使う目的は，もっぱらクラッシュからの回復にある．各要求が他の要求から独立しているならば，クラッシュ回復の手続きを行わず，クライアントやサーバを失

Courtesy of NAS

```
Script started on Wed Nov 28 13:52:41 1990  (Cray YMP)
(/dev/ttyp105)
reynolds.lkoke 31 df
/navier/usr    (navier-uip:/usr  ):     21772  4K blocks (  7.3%)
/navier/scr4   (navier-uip:/scr4 ):     797848  4K blocks ( 30.9%)
/navier/scr3   (navier-uip:/scr3 ):     1706166  4K blocks ( 66.1%)
/navier/scr2   (navier-uip:/scr2 ):     1051558  4K blocks ( 40.7%)
/navier/scr1   (navier-uip:/scr1 ):     1518537  4K blocks ( 58.8%)
/navier/u/ng (navier-uip:/u/ng ):     7802  4K blocks (  2.6%)
/navier/u/ne (navier-uip:/u/ne ):     222234  4K blocks ( 24.9%)
/navier/u/nd (navier-uip:/u/nd ):     301989  4K blocks ( 50.8%)
/navier/u/nc (navier-uip:/u/nc ):     268600  4K blocks ( 30.1%)
/navier/u/nb (navier-uip:/u/nb ):     114466  4K blocks ( 13.1%)
/navier/u/na (navier-uip:/u/na ):     482460  4K blocks ( 54.1%)
/proc          (/proc ):                108297  4K blocks ( 42.7%)      566 procs
/51root/usr/src (/dev/dsk/src ):        130113  4K blocks ( 29.2%)
/51root/usr    (/dev/dsk/usr ):         51508  4K blocks ( 17.3%)
/51root        (/dev/dsk/root ):        108252  4K blocks ( 72.6%)
/usr/src       (/dev/dsk/60src ):       156279  4K blocks ( 38.0%)      18537 I-nodes
/scr8          (/dev/dsk/scr8  ):       644989  4K blocks ( 25.1%)
/scr7          (/dev/dsk/scr7  ):       1169864  4K blocks ( 45.5%)
/scr6          (/dev/dsk/scr6  ):       567802  4K blocks ( 22.1%)
/scr5          (/dev/dsk/scr5  ):       1822596  4K blocks ( 70.9%)
/u/rf          (/dev/dsk/rf     ):      161337  4K blocks ( 27.1%)
/u/re          (/dev/dsk/re     ):      627650  4K blocks ( 52.8%)      92220 I-nodes
/u/rd          (/dev/dsk/rd     ):      834365  4K blocks ( 70.2%)      110770 I-nodes
/u/rc          (/dev/dsk/rc     ):      567311  4K blocks ( 47.7%)      91135 I-nodes
/u/rb          (/dev/dsk/rb     ):      86984  4K blocks (  9.7%)       82841 I-nodes
/u/ra          (/dev/dsk/ra     ):      820870  4K blocks ( 69.1%)      105061 I-nodes
/core          (/dev/dsk/core ):        351213  4K blocks ( 78.7%)      65525 I-nodes
/usr/spool     (/dev/dsk/spool ):       42737  4K blocks ( 37.3%)       27943 I-nodes
/wrk           (/dev/dsk/wrk   ):       708035  4K blocks ( 59.6%)      166903 I-nodes
/usr           (/dev/dsk/60usr ):       59623  4K blocks ( 20.1%)       18223 I-nodes
/tmp           (/dev/dsk/tmp  ):        586476  4K blocks ( 98.7%)      65046 I-nodes
/              (/dev/dsk/60root  ):     100229  4K blocks ( 67.2%)

reynolds.lkoke 33 /etc/mount
/  on          /dev/dsk/60root read/write on Mon Nov 26 06:24:52 1990
/tmp on        /dev/dsk/tmp    read/write,rw on Mon Nov 26 06:25:03 1990
/usr on        /dev/dsk/60usr  read/write,rw on Mon Nov 26 06:25:04 1990
/wrk on        /dev/dsk/wrk    read/write,rw,quota=/etc/quota60/wrk
                                   on Mon Nov 26 06:25:05 1990
/usr/spool on /dev/dsk/spool  read/write,rw,quota=/etc/quota60/spool
                                   on Mon Nov 26 06:25:06 1990
/core on       /dev/dsk/core   read/write,rw on Mon Nov 26 06:25:06 1990
/u/ra on       /dev/dsk/ra     read/write,rw,quota=/etc/quota60/home
```

6-6 CRAY YMP上のファイルシステム

```
                                                on Mon Nov 26 06:25:07 1990
/u/rb on              /dev/dsk/rb       read/write,rw on Mon Nov 26 06:25:08 1990
/u/rc on              /dev/dsk/rc       read/write,rw,quota=/dev/dsk/ra
                                                on Mon Nov 26 06:25:09 1990
/u/rd on              /dev/dsk/rd       read/write,rw,quota=/dev/dsk/ra
                                                on Mon Nov 26 06:25:10 1990
/u/re on              /dev/dsk/re       read/write,rw,quota=/dev/dsk/ra
                                                on Mon Nov 26 06:25:10 1990
/u/rf on              /dev/dsk/rf       read/write,rw on Mon Nov 26 06:25:11 1990
/scr5 on              /dev/dsk/scr5     read/write,rw,quota=/etc/quota60/scratch
                                                on Mon Nov 26 06:25:12 1990
/scr6 on              /dev/dsk/scr6     read/write,rw,quota=/dev/dsk/scr5
                                                on Mon Nov 26 06:25:12 1990
/scr7 on              /dev/dsk/scr7     read/write,rw,quota=/dev/dsk/scr5
                                                on Mon Nov 26 06:25:13 1990
/scr8 on              /dev/dsk/scr8     read/write,rw,quota=/dev/dsk/scr5
                                                on Mon Nov 26 06:25:14 1990
/usr/src on           /dev/dsk/60src    read/write,rw on Mon Nov 26 06:25:14 1990
/51root on            /dev/dsk/root     read/write,rw on Mon Nov 26 06:25:15 1990
/51root/usr on        /dev/dsk/usr      read/write,rw on Mon Nov 26 06:25:16 1990
/51root/usr/src on    /dev/dsk/src      read/write,rw on Mon Nov 26 06:25:16 1990
/proc on              /proc             read/write on Mon Nov 26 06:25:17 1990
/navier/u/na on       navier-uip:/u/na read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:18 1990
/navier/u/nb on       navier-uip:/u/nb read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:19 1990
/navier/u/nc on       navier-uip:/u/nc read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:19 1990
/navier/u/nd on       navier-uip:/u/nd read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:20 1990
/navier/u/ne on       navier-uip:/u/ne read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:21 1990
/navier/u/ng on       navier-uip:/u/ng read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:21 1990
/navier/scr1 on       navier-uip:/scr1  read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:21 1990
/navier/scr2 on       navier-uip:/scr2  read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:22 1990
/navier/scr3 on       navier-uip:/scr3  read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:22 1990
/navier/scr4 on       navier-uip:/scr4  read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:23 1990
/navier/usr on        navier-uip:/usr   read only,nosuid,bg,soft
                                                on Mon Nov 26 06:43:23 1990
reynolds.lkoke 34 exit
reynolds.lkoke 35
```

Courtesy of NAS

6-6 (続き) CRAY YMP上のファイルシステム

っても問題はない．

内部状態がないということは，もし回復作業が必要であれば，その操作を行うのはクライアントの責任であるということである．サーバがクラッシュしてその後回復した場合でも，NFS 自身はバックアップを始めるために何の回復手続きも行う必要はない．

NFS プロトコルが提供する基本操作は，単にローカルファイルシステムで求められるサービスを拡張したものにほかならない．図 6-7，図 6-8 は基本的な検索操作を示したものである．あるクライアントがファイル名を送り出し，そのファイル名について検索を要求する．検索の結果，ファイルハンドルが返される．これは，以降の読出し，応答操作で，対象を識別するために使われる基本通貨となる．

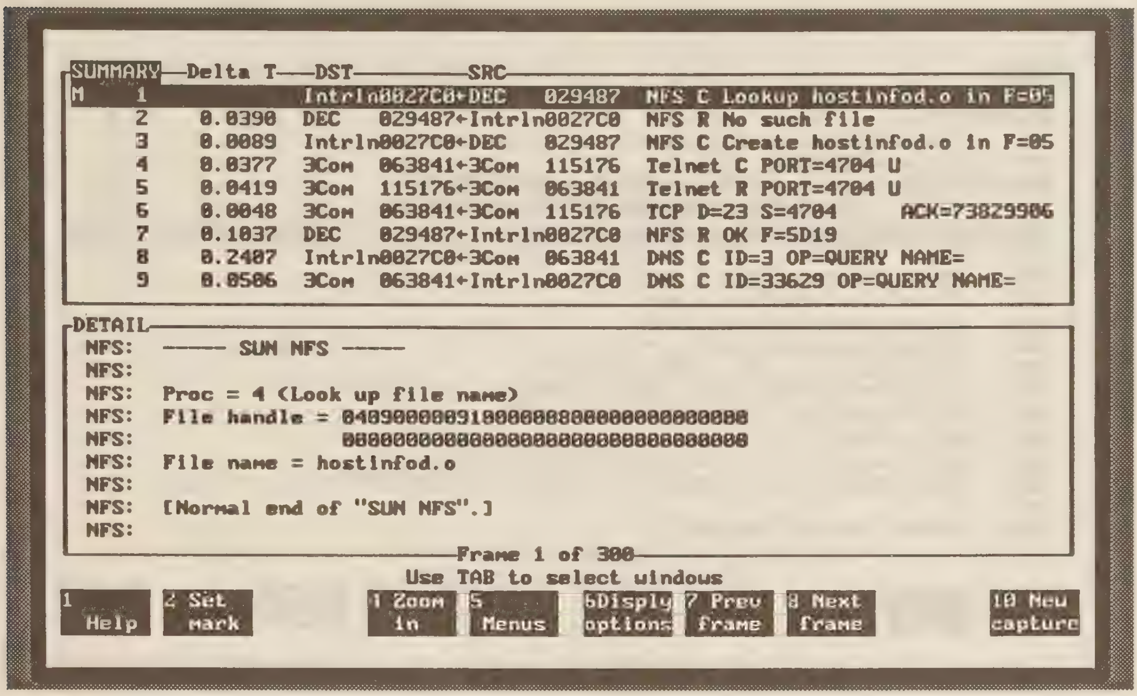
検索コマンドに対する応答はどのような場合でも検索応答である．この応答はファイルに付けられている様々な属性値をユーザに対して送り返す．図 6-9 は getkey.c というファイルに対する検索要求を詳しく示したものである．ファイルの属性値は非常に Unix のものと似ていて，Unix ファイルについている許可モードビットも含まれている点に注意してほしい．

ファイルに対して返される属性値の中で重要なのは変更時刻である．このデータはキャッシュされているファイルがサーバ上のものと同じバージョンかどうかを判断するために使われる．ファイルシステムがファイルを受け取ると，そのファイルをユーザのために変更時刻とともにキャッシュしておくことが多い．ユーザがそのデータを変更したいといってきた場合には，ワークステーションはキャッシュの中の変更時刻と，サーバ上の変更時刻を比較すれば，その間に他の誰かが変更を加えたかどうかを判断することができる．

ファイルに対する検索は，普通のファイル以外の情報を返してくることもある．例えば図 6-10 では，検索コマンドに対して，そのファイルは実は他のファイルへのシンボリックリンクであるということを示す属性値が返されている．シンボリックリンクがクライアントのカーネルに返されると，カーネルは普通，他のファイルへのリンクをたどるためにリンク読出しコール (read-link call) を発行する (その結果がまたリンクということもあり得る)．このようにして本当のファイルにたどり着くまで，リンクがたどられる．

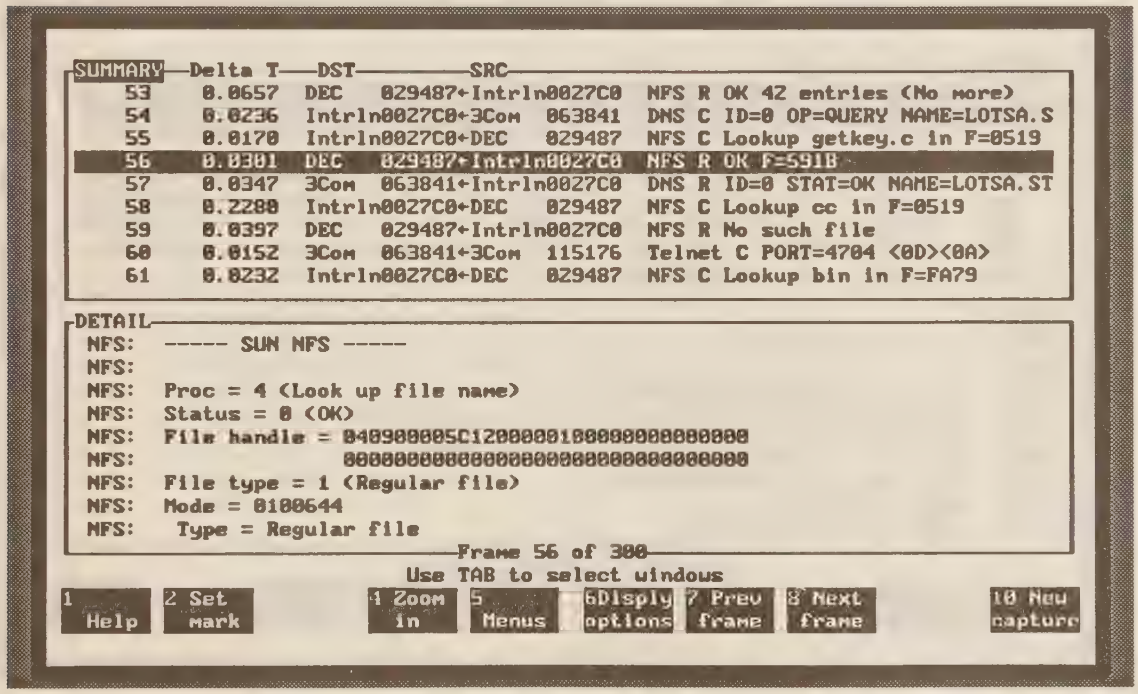
図 6-11 と図 6-12 は，また別の種類の検索操作を示している．図 6-11 ではワークステーションが NFS サーバに属性値読出しコール (get attributes call) を発行している．次のパケットを見ると“成功”という応答が返っていることがわかる．それからクライアントはこれに続けてディレクトリの中にある特定のファイルに対してもう 1 回検索を行っている．

検索コマンドを実行すると，ファイルへのファイルハンドルが返され，クライアントシステムはこれを使って読出し要求を発行することができるようになる．図 6-13 は読出し要求を示す．このパケットは 3000 バイトあまりのデータを含む NFS コールであることがわかる．パケットの最後には，そのパケットには 1360 バイトのデータが入っているが，1712 バイトはここには存在しない (つまり，次のパケットでこれから来る)，ということが示されている．



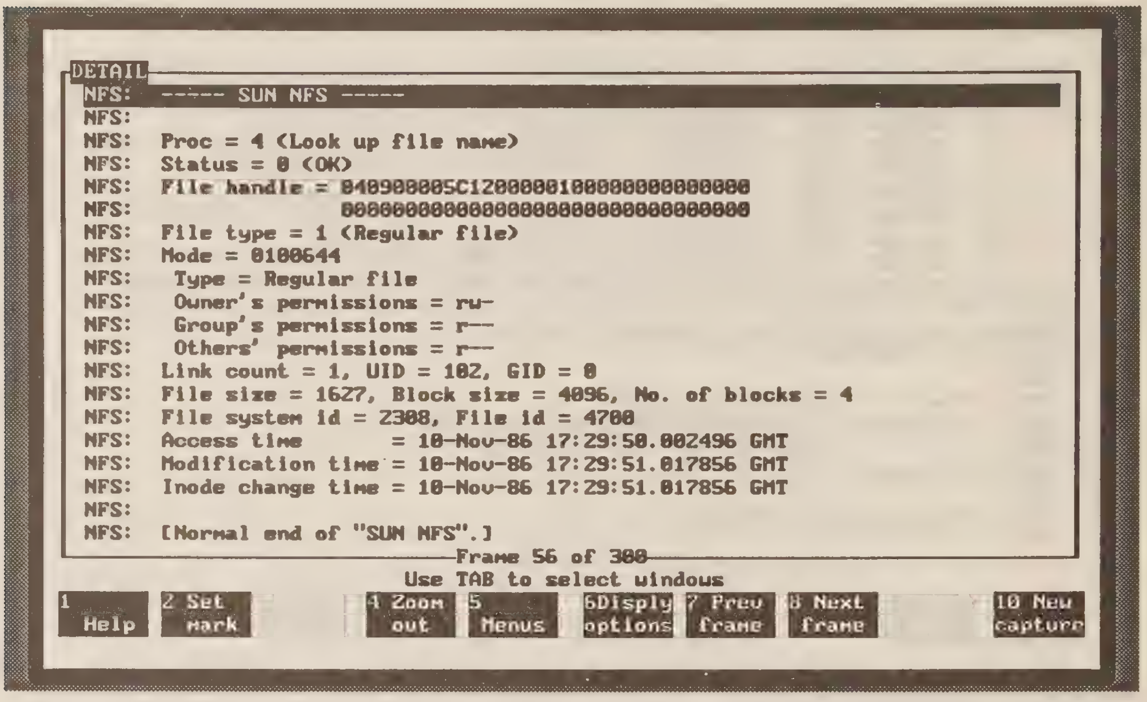
Courtesy of Network General

6-7 NFS検索操作



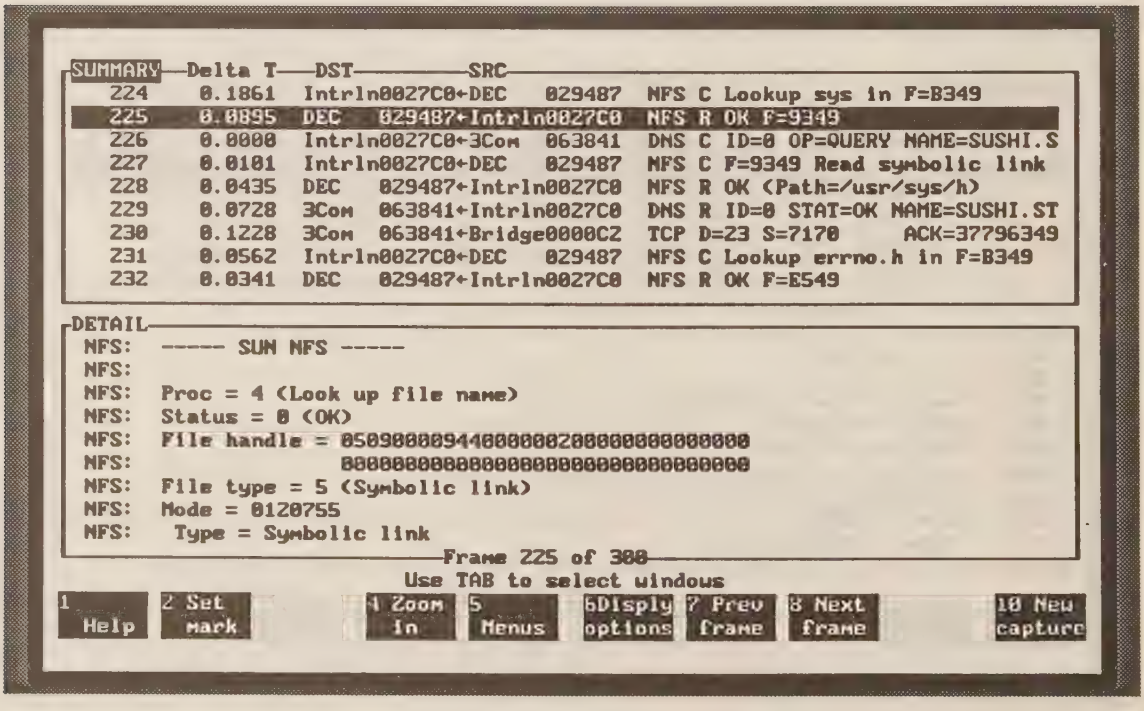
Courtesy of Network General

6-8 検索トラフィック



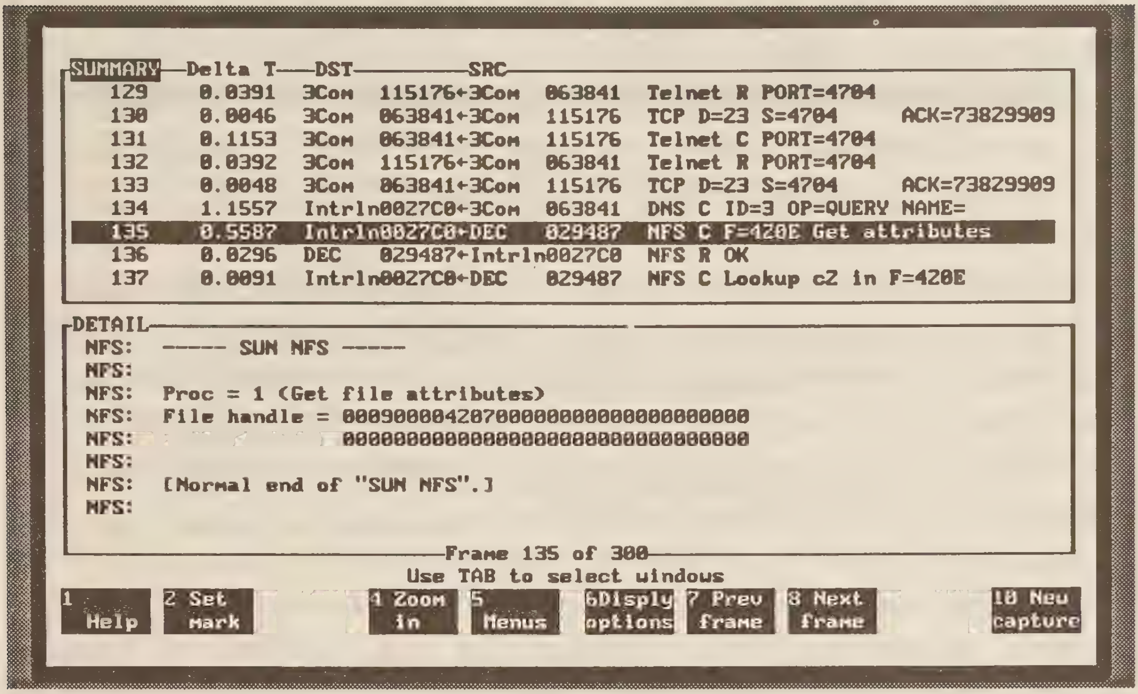
Courtesy of Network General

6-9 検索に対する応答



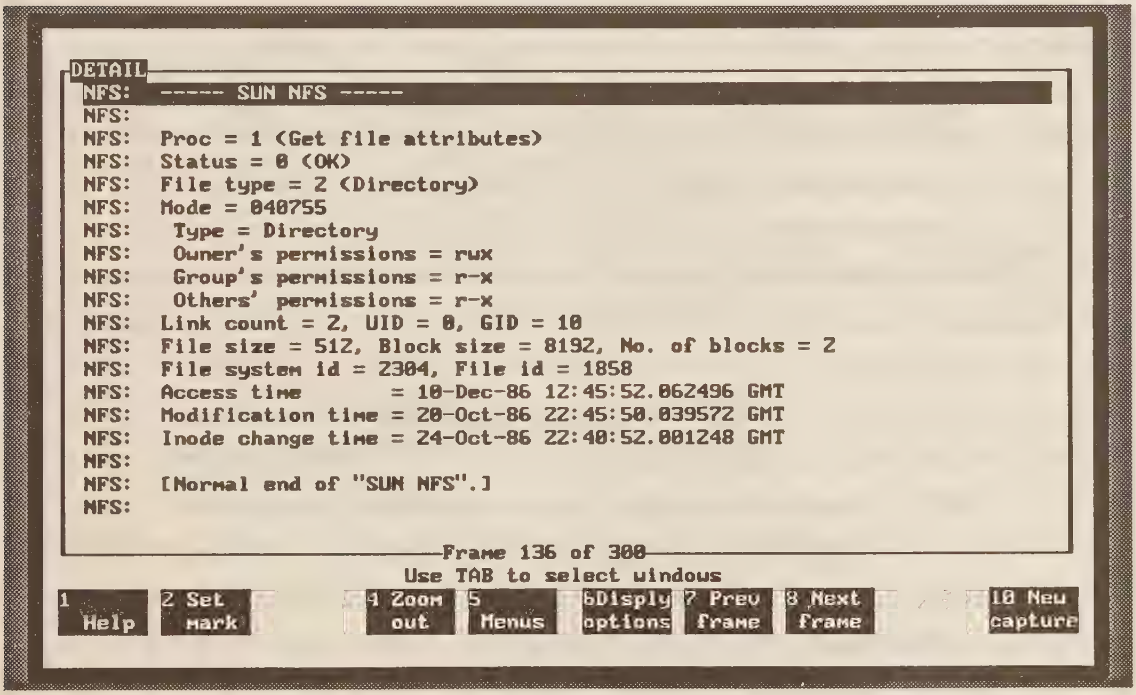
Courtesy of Network General

6-10 シンボリックリンクの追跡



Courtesy of Network General

6-11 属性読出し要求



Courtesy of Network General

6-12 NFS属性読出し応答

NFS は RPC コールを使ってネットワーク上のデータのやりとりをしている．さらに RPC は基礎として UDP や TCP 通信サービスを使っている．Sun の UDP の実装ではメッセージの長さは 8 キロバイトまでしか許されないが，TCP ではストリーム風インターフェイスによって無限長の“メッセージ”が許される(TCP のパケットには 64 キロバイトまでという制限はあるが，TCP へのインターフェイスは単なるデータのストリームである)．

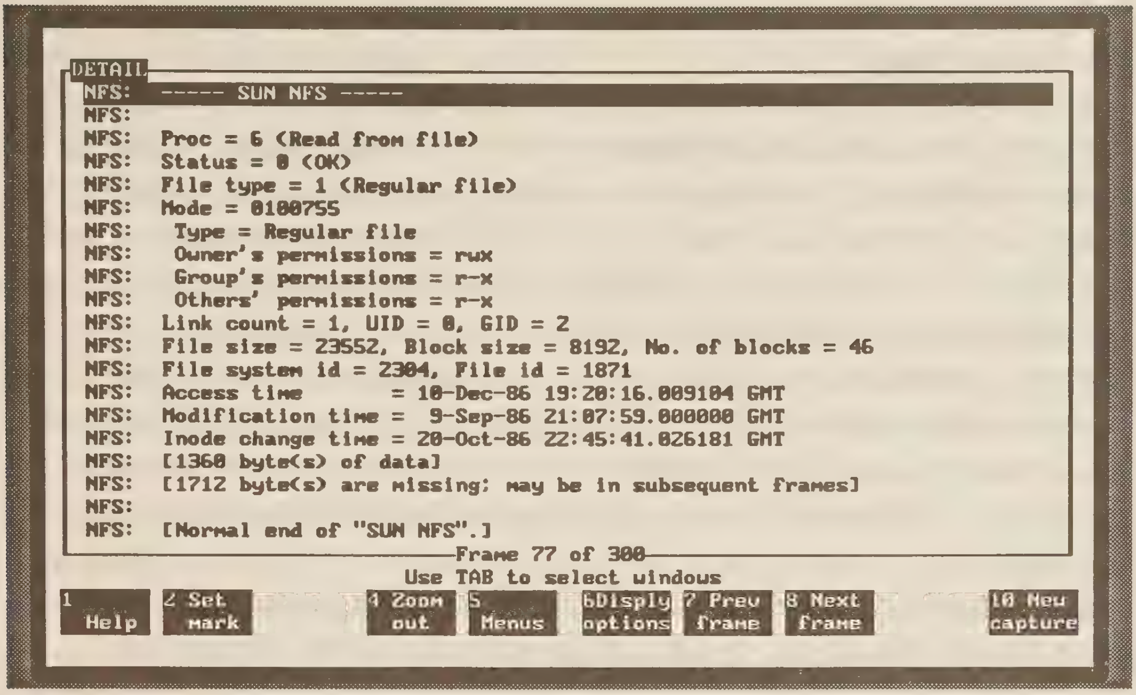
図 6-14 はその「まだ来ていないデータ」がどうなったか，ということを示している．最初の NFS パケットが送られたあとに，UDP コンティニュエーション(continuation)が 2 つ続いていることがわかる．これらの UDP コンティニュエーションには残りのデータが保持されており，RPC はこれらを用いて 1 つの応答の形にまとめ，ワークステーションやサーバに対して完全な応答として渡す．UDP コンティニュエーションはそれより上の階層のユーザには完全に透明である――UDP は送られてくる IP フラグメントを完全なパケットに組み直す(もっとも，最善をつくすという方針でやっているだけで，到着にはなんの保証もない)．

UDP は 8 キロバイトのデータグラムサービスという外見をとるが，その基盤となるネットワークの最大転送単位(maximum transfer unit : MTU)は重要である．UDP パケットがフラグメントに分割され，その中のフラグメントの 1 つが失われてしまった場合には UDP パケット全体が失われることになる．つまり上の階層のユーザ(この場合 UDP)は，たとえ失われたのが 576 バイトの IP パケットであっても，8 キロバイトのデータを再転送する．NFS が広域ネットワーク環境で使われるようになった場合には，リンクがより複雑で信頼度も低くなるので，トランスポート層の提供するサービスや，基盤となるネットワークの MTU がますます重要になってくる．

NFS で TCP に基づいたトランスポートサービスを使うことの利点の 1 つは，TCP が到着の保証を提供していることである．(この場合)データの再送は RPC のレベルでなく，TCP のフラグメントのレベルで行われる．TCP は基盤ネットワークの性能に適応できるので，混雑している設備を UDP よりも効率よく利用することができる．

NFS も基盤ネットワークをまったく無視しているわけではないことに注意してほしい．適応型再送法(adaptive retransmission method)を使って，要求がタイムアウトするたびに盲目的に再試行をしたりはしないようにしたクライアントの実装も多い．再試行ごとに時間間隔を増やしていくことによって，ネットワーク混雑あるいはネットワーク障害時の再送によるネットワーク負荷の軽減をはかるのである．この適応型再送法ではパケットサイズの調節も採り入れられており，ネットワーク資源が不足しているときには小さなパケットを送るようにしている．

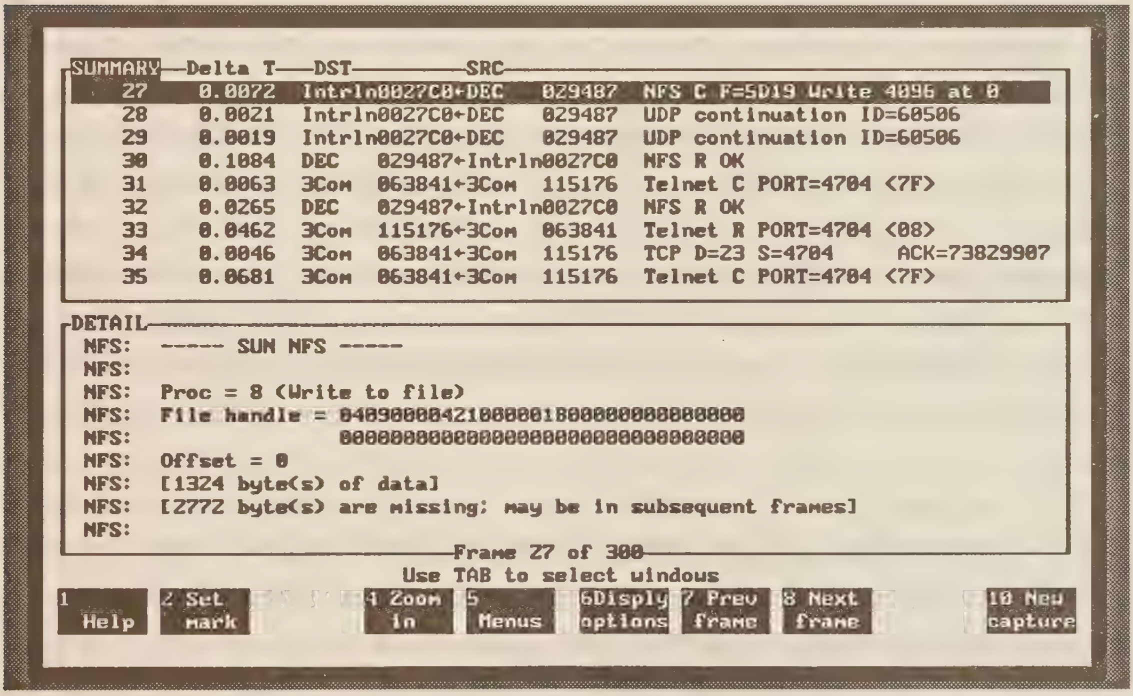
クライアントの中には適応型再送法を実装していないものもある．この場合，クライアントは単に再試行間隔分だけ待って再試行を行う．再試行間隔はデフォルトでは 0.7 秒である．それからクライアントは現在の間隔を 4 倍してもう 1 度再試行して，最終的に 60 秒の上限に至ったら止める．ある与えられたタイムアウトサイクルの中で何回再試行するかはマウントポイントごとに設定できる．デフォルトでは 3 回である．



```
DETAIL
NFS:   ----- SUN NFS -----
NFS:
NFS:   Proc = 6 (Read from file)
NFS:   Status = 0 (OK)
NFS:   File type = 1 (Regular file)
NFS:   Mode = 0100755
NFS:    Type = Regular file
NFS:    Owner's permissions = rwx
NFS:    Group's permissions = r-x
NFS:    Others' permissions = r-x
NFS:   Link count = 1, UID = 0, GID = 2
NFS:   File size = 23552, Block size = 8192, No. of blocks = 46
NFS:   File system id = 2304, File id = 1871
NFS:   Access time       = 10-Dec-86 19:20:16.009104 GMT
NFS:   Modification time =  9-Sep-86 21:07:59.000000 GMT
NFS:   Inode change time = 20-Oct-86 22:45:41.026181 GMT
NFS:   [1360 byte(s) of data]
NFS:   [1712 byte(s) are missing; may be in subsequent frames]
NFS:
NFS:   [Normal end of "SUN NFS".]
                          Frame 77 of 300
                     Use TAB to select windows
1      2 Set        4 Zoom  5       6Disply 7 Prev  8 Next           10 New
 Help    mark         out    Menus  options  frame   frame           capture
```

Courtesy of Network General

6-13 NFS読出し操作



```
SUMMARY  Delta T   DST              SRC
    27   0.0072  Intrln0027C0+DEC    029487  NFS C F=5D19 Write 4096 at 0
    28   0.0021  Intrln0027C0+DEC    029487  UDP continuation ID=60506
    29   0.0019  Intrln0027C0+DEC    029487  UDP continuation ID=60506
    30   0.1084  DEC    029487+Intrln0027C0  NFS R OK
    31   0.0063  3Com   063841+3Com   115176  Telnet C PORT=4704 <7F>
    32   0.0265  DEC    029487+Intrln0027C0  NFS R OK
    33   0.0462  3Com   115176+3Com   063841  Telnet R PORT=4704 <08>
    34   0.0046  3Com   063841+3Com   115176  TCP D=23 S=4704      ACK=73829907
    35   0.0681  3Com   063841+3Com   115176  Telnet C PORT=4704 <7F>
DETAIL
NFS:   ----- SUN NFS -----
NFS:
NFS:   Proc = 8 (Write to file)
NFS:   File handle = 04090000421000001B00000000000000
NFS:                 00000000000000000000000000000000
NFS:   Offset = 0
NFS:   [1324 byte(s) of data]
NFS:   [2772 byte(s) are missing; may be in subsequent frames]
NFS:
                          Frame 27 of 300
                     Use TAB to select windows
1      2 Set        4 Zoom  5       6Disply 7 Prev  8 Next           10 New
 Help    mark         in     Menus  options  frame   frame           capture
```

Courtesy of Network General

6-14 UDPコンティニュエーションを伴ったNFS書込み

6.3 NFSの実装

NFSのサービスを提供しているのは，NFSサーバデーモン(NFSD)である．1つのUnixシステムにはこれを数個もつことができる．システムに8個のデーモンがあるならば，同時に8つの要求を処理することができる．Unixクライアントには，オプションで(汎用に実装されてはいるが)ブロックI/Oデーモン(BIOD)がある．これらのデーモンは，プログラムが要求する前にデータの読込み(非同期先読み)，また遅延書込みを行う．

BIODでの遅延書込みとは，プログラムがファイルに書込みをしたときに，クライアントプログラムのwriteコマンドはデータがキャッシュに書き込まれるとすぐに戻ってくるということである．この時点でデータはまだサーバには存在しない．しかし，プログラムがファイルに対してクローズすると，その呼出しはすべてのデータが安定な記憶領域に吐き出されきるまで戻ってこない．

BIODの数はクライアントとサーバの性能にかなり影響する．BIODが多ければ，ある程度まではクライアントの性能は上がる．特に書込み操作が頻繁に行われる場合には有効である．

6.3.1 状態の付加

NFSプロトコルに内部状態がないことは，クラッシュからの復帰には都合がよい．しかし，何か内部状態があった方が，性能向上に役立つこともある．DECとSunで行われた研究では，若干の内部状態をサーバに加えてやることで劇的な性能の向上が実現された．

基本的なアイデアは，サーバに余計な仕事をさせてしまう重複したNFS要求を発見することである．要求の重複はサーバが遅かったり，ネットワークの混雑(それによって応答が失われる)が原因となって生じる遅れによって引き起こされる．内部状態のない環境では，べき等操作(idempotent operation)は再度行ってよい．しかしべき等でない要求は，一般に非破壊的に再実行することはできない．読出しはべき等要求の例である．あるファイルブロックを繰り返し読み出してもなんら悪影響はない．create(ファイル作成)は再実行するとエラーになるのでべき等ではないとみなされる．こういった要求の重複を防ぐことは，性能の向上という都合のよい結果をもたらす．

サーバが遅い場合，クライアントは再送のためにいくつも要求を送るかもしれない．これはよくてサーバの資源を消費し，悪くすれば間違った結果を招く．NFSの書込みはNFSサーバの操作の中で最もコストの高いものである．サーバが書込みで過負荷状態にあるときには，得てして重複した書込み要求を処理するというダブルパンチをくらってしまう．

1つのプロセスにいくつかの未処理の読出し，書込み要求があることがある．特にクライアントでBIODを動かしているときにはそうである．これらのプロセスはクライアントの先行読出し，

遅延書込み機能をクライアントの他のプロセスに代わって行っている．それぞれの要求は，独立にタイムアウトして再送が行われる．

再送の問題についてサーバの側で管理者ができることはあまりない．サーバの負荷が高くなったときに，要求の負荷を制御するのはクライアント側に頼らざるを得ない．しかしそもそもクライアントには思いやりがあることは期待されていない．ユーザにはいくつでも好きなだけ BIOD を走らせることが許されている場合が多いし，どんな短い再送タイムアウト値でマウントするのも自由である，というのが一般的なクライアントの実装である．

NFS 第 2 版の 16 個の操作のうち，9 個はべき等でない．すなわち create, remove, link, symlink, mkdir, rmdir, rename, setattr, write である．最初の 7 つはまったく自明であろう．1 度目でファイルが作成されると，2 度目のその create 操作の実行は（“ファイルはすでに存在する（file already exists）”というメッセージを出して）失敗する．最後の 2 つ，set attribute と write はさほど自明ではない．時間をあけないで実行されるならべき等であり，同じ結果になる．

しかし，間に別の操作が入って再実行されたとしたらどうなるだろうか？ この場合結果はちょっと妙なことになる．例えば set attribute コマンドは，ファイルを切り捨てるために使うことができる．一度ファイルを切り捨てて，他の誰かがそれにデータを付け加えたとしよう．それから 2 回目の切捨てが行われる．

重複した要求を防ぐ機能はもともとのカーネル RPC 参照実装（reference implementation）の一部分であり，これは最近処理したトランザクション ID（xid）のキャッシュを保持している．副作用のある操作のトランザクション ID がキャッシュに保存される．操作が成功した場合にはキャッシュに問い合わせる必要はない．しかし操作が失敗した場合には，キャッシュをチェックして失敗が重複したトランザクションによるものかどうかを調べる．この標準的な動作はすべての NFS 実装に見られるものである．create, remove, link, mkdir, rmdir が失敗するとキャッシュをチェックする．

この重複した ID をチェックする操作は，2 種類の変更がなされた．第 1 に，すべての操作を含むように拡張された．第 2 にチェックは操作の後ではなく，前に行われるようになった．

DEC の Chet Juszczak はこの変更作業を行い，変更の実装について 2 種類の VAX サーバによる実験を文書で報告している．それによると，小さいサーバの方が重複した書込みをより受けやすいということがわかった．処理能力を越えてしまいやすく，その結果，十分な速さで応答できなくなりがちだからである．

活発なクライアント（MicroVAX II）と小型サーバ（これも MicroVAX II）で，1 メガバイトのデータを 8 キロバイトの要求 128 個によってファイルに書き込んだ．ファイルは書込みが始まった時点ではいつも空である．小さいサーバの場合には，クライアントは 190 の書込み要求を発生して処理を終えた．このうち 62 が重複している．サーバはこの余分な 62 の書込み要求のうち 61 を処理し，実効書込み速度は 35 キロバイト/秒であった．

大きなプロセッサでは，サーバ処理能力に余裕があるので，重複した書込み要求はより少なかった．例えばVAX8550では141しか書込み要求が発生せず，13しか重複しなかった．実効書込み速度は20回の繰返しで70キロバイト/秒にものぼった．

次に，RPCトランザクションキャッシュでキャッシュするトランザクションの範囲を増やし，すべての種類の要求を加えてみた．各キャッシュのエントリには“処理中(in progress)”フラグが付け加えられ，トランザクション完了コード，タイムスタンプも加えられた．キャッシュの使い方も異なる．新しい処理法では，NFS要求に応える一番最初の動作の1つでキャッシュのチェックを行う．要求がキャッシュの中にあり，処理中という印が付いている場合には，要求は単に捨てられ何の応答も返さない．要求がキャッシュの中にない場合は(キャッシュに)加えられ，処理中と印が付けられる．

トランザクションがキャッシュの中にあるが処理中ではない場合は，働きが少し違っている．重複が廃棄ウィンドウ(普通3から6秒)以内に起こった場合には，パケットは捨てられ何の応答も返さない．廃棄ウィンドウ以降であればサーバはその要求を実行する．これは最初の応答が，再送された要求と行き違いになるというよくある問題を解決するのが目的である．

最後に，べき等でない呼出しであるsetattrとwriteを特別扱いするようにした．これらの操作については操作が終了したときの完了ステータスとタイムスタンプが保存される．操作が成功して終了した場合にはキャッシュのエントリが更新される(ファイルが最後に変更された時刻を表すタイムスタンプの変更も含む)．すでに成功している要求を受け取ったとき，タイムスタンプ以来ファイルが変更されていなければ，成功の応答がクライアントに送られるが，実際には操作を行わない．しかしもし重複した要求を受け取った時点でキャッシュのエントリがすでに掃き出されていた場合には，その操作は再度実行される．

さて，これらの改良の結果はどうだろうか? 重複した書込み要求はすべて消去され，処理されることはなくなった．小型のサーバでも書込み速度が35キロバイト/秒から48キロバイト/秒になり，大型のサーバでは75から83キロバイト/秒になった．

ここで重要なのは，キャッシュがうまく働いたということよりも，プロトコルとしては内部状態がないという性質を保ちつつ，サーバに内部状態が加えられたという事実である．内部状態のないプロトコルは，クラッシュからの復帰が非常に簡単であることを意味する．重複要求をチェックすることは高負荷時の性能が向上したというだけでなく，データの一貫性もさらによく保たれるようになったということを意味する．

6.3.2 状態

内部状態のないプロトコルであることを保ちつつNFSの実装に保持させることのできる内部状態情報には，この他にもいろいろなものがある．サーバにあるファイル中のデータも永続的な(persistent)状態にほかならないということは明らかであろう．この内部状態は悪いものではな

い．これがあってこそ NFS を使う意味があるのだから．NFS に内部状態がないというのは，プロトコルとサーバにこれ以外の制御状態がないということをいうのである．例えばクライアントには，対象を得るために必要なサーバのファイルハンドルがキャッシュされている．ファイルハンドルをキャッシュするということは，クライアントは親ディレクトリにある対象を再検索させられることはないということである．

クライアントはデータと属性のキャッシュも保持することができる．クライアントにある BIOD は非同期先読みを行って，クライアントが実際に要求してくる前にデータをとってきている．キャッシュの中のこのデータはどんどん古くなるから，プログラムはそれを再度書き出す前にデータが同じであるかどうかを常にチェックしている．

サーバも性能を上げるために状態を保持している．状態をもった操作の代表的なものは先読み(read-ahead)キャッシュである．サーバに状態を付け加えることはシステム依存であり，NFS プロトコルには規定はない．

資源をどう公開するかについてはサーバ側に自由度がある．管理者はマシンやネットグループ(マシンのグループ)のリストにアクセス権を割り当てることができる．また管理者はクライアントに DES 認証を要求することもできる(secure NFS)．

ファイルへのアクセスはそもそも状態をもつものだということに注意してほしい．サーバがどうにか状態をもたないでいられるのは，操作を実行するために必要な状態（の情報）をクライアントに送り返しているからである．クライアントにファイルハンドルを送り返すのはその典型的な例である．クライアントはそのハンドルを隠蔽(opaque)データ構造として扱う．つまりサーバは，何であれ必要な状態の情報をハンドルの中に隠すことができるわけである．

このようなファイルハンドルの一例は SunOS ベースのサーバが返すハンドルである．この場合 3 つの情報が入っている．すなわちファイルシステム ID，i ノード番号，世代番号(generation number)である．

6.4 MVS オペレーティングシステム上の NFS

NFS プロトコルが応用できるのは何も Unix 環境だけには限らないということを示すためには，IBM の MVS オペレーティングシステム上の NFS の実装を見てみるのがよいであろう．IBM の MVS オペレーティングシステムでユーザアプリケーションを多く実行させるのは，いわばピザを配達するのにトラックを使うようなものである．しかし，IBM のメインフレームの能力が大規模な組織ネットワークと実にうまくいく例もあるのである．IBM のメインフレームは非常にすぐれたファイルサーバとなり，大容量のトランザクション処理を実にうまく扱う．

メインフレームはデータの入れ物(vault)としては完璧であるが，端末には IBM3270 よりもワークステーションの方がよいであろう．事実これは，まさしく Sun 社が最も重要なデータを扱う

ために採っている方法である．製造用の購買 MRP システムは，IBM 互換のメインフレーム上に保存されている．SNA ゲートウェイの製品 (16 章を参照) を使うことにより，ワークステーションから MVS のファイルシステムをマウントして組織データにアクセスすることができるのである．

MVS/NFS の実装は Sun 社と Electric Data Systems 社(EDS)の共同で開発された．その実装では MVS はサーバ専用である．メインフレームに IBM/PC のディスクをマウントさせるようなサービスは，これでは提供されない．

MVS/NFS 実装の基本的な目標は，MVS のデータを Sun あるいは他の NFS クライアントから利用できるようにすることである．クライアントは IBM/PC，VAX，Sun，Macintosh などが考えられる．これらの 2 つの環境の橋渡しはゲートウェイを介して行われる．Sun のネットワークから見ると端点はゲートウェイであり，メインフレームから見るとチャネルに付加されたフロントエンドプロセッサ(FEP)が端点である．

メインフレームへのアクセスは Sun ベースの FEP を使って実現されている．このソフトウェアには TCP/CTCA の変換機構と RPC 呼出しをメインフレームへと転送するプログラムが含まれている．この FEP は単に Sun ワークステーションに SunLink チャネルアダプタとソフトウェアをのせたものである．FEP は他のワークステーションとイーサネット上に同居することもできるし，インターネットルーティングを使って WAN 回線上にサービスを提供することもできる．RPC 転送機構はソケットの作成とサービスの登録をメインフレーム上のサーバのために行い，RPC パケットをネットワークからメインフレーム上の登録されたサービスに送る経路制御を行う．

NFS サービスにアクセスするクライアントは mvsmount という (普通とは) 違ったマウントプロトコルを使う．普通のマウントプロトコルが使えないのは，FEP が間に入っているためと，MVS サーバにログインしてパスワードを送らなければならないためである．

MVS の RPC ライブラリは RPC の標準的な実装である．どんなアプリケーションでも RPC インターフェイスを使うように書ける(あるいは移植できる)．既存のアプリケーションの 1 つが NFS である．性能の向上と保安上の理由から，NFS は他のアプリケーションから離され別個の MVS タスクになっている．

ゲートウェイ側では RPC 転送機構が大量の作業を行う．メインフレームでこれにあたるのはチャネルタスクである．このタスクはチャネルハードウェアアダプタを通じて RPC 転送機構と通信している．この 2 つの間で特別なプロトコルを使用することで，サービスの登録と登録抹消，チャネルリセット機能，複数の RPC パケットの一括ブロック化，割込みとチャネル設定のオーバーヘッドを減らすための情報の制御などが可能になる．

6.4.1 NFS サーバタスク

NFS サーバタスクはマルチタスクかつマルチスレッド化されている．つまり，共有されたコードと独立したデータとスタック領域からなる NFS 処理タスクの複数のコピーが，単一のアドレス空間で並行に実行できる．

ここでユーザと実装者にとって問題になるのは，NFS と MVS のファイルシステムが非常に異質なものであるということである(図 6-15)．MVS はレコード指向のデータセットで EBCDIC 符号を使っている．他方，NFS はバイト列であってレコード構造はなく，文字セットはデフォルトでは ASCII である．

違いはこれだけではない．MVS はデータセットごとに属性がついているが，属性の大半は NFS の属性に対応するものがないうえ，データセットの種類により属性のタイプも変わってくる．さらに面白いことに，インストールされたソフトウェアのタイプによっても属性が変わるのである．例えば，アクセス制御機構第 2 版(Access Control Facility 2 : ACF2)をインストールすると属性は増える．

他にも大きな相違点がある．NFS は非常に強力な階層化ファイルシステムの概念をもっているが，MVS には，データセットが平面的なファイル空間に積まれているだけの非常に弱い階層性しかない．

MVS はレコード方式のファイルシステムなので，テキストの各行をレコードとして区切っている．NFS の世界では，サーバ側ではテキストもバイナリも違いはなく，ファイルの内容の解釈はクライアントに任されている．さらに，MVS はキーによるアクセスができるが，これは NFS にはない機能である．

レコードとファイルの間の違いはアクセスだけに限ったことではない．ファイルの大きさが実際違うかもしれないのである．MVS のデータセットではファイルの大きさはすぐにはわからない．ファイルのタイプによってはレコードの数（もっとひどいときにはディスクのトラック数）だけしか返ってこないこともある．それがわかっても，今度はブロックと部分ブロックへのレコ

MVS	NFS
レコード指向	バイト列
複雑なルールベースの安全管理	ユーザ/グループ/ワールド許可
EBCDIC	ASCII
弱いファイル階層	ファイルシステムに階層が内在
ファイル階層への操作のとても弱い対応づけ	明確に定義された階層への操作
テキスト行をレコードとして区切る	テキスト/バイナリはクライアントが解釈
構造ファイルを定義	キーでアクセスするファイルはサポートしない

6-15　MVSファイルシステムとNFSファイルシステム

ード詰込みという問題があり，大きさが返されたとしても，それが正しいとは限らない．

これが問題となるのは，NFSではバイトオフセット（先頭からのバイト数）と，必要とするブロックの長さを指定してファイルの内容を参照するからである．ファイルのサイズや全レコードの場所がわからないとNFSの要求を満たすことは難しい．最悪の場合，NFSサーバはただ単に属性値読出し要求に応えるだけのためにファイルを全部読んでオフセットを計算しなければならなくなる．

レコード構造のために，データセットの読み方についての問題も起こってくる．バイナリとテキストで切り替えなければならないということである．バイナリの場合，MVS/NFSはデータ列の内容には何の変換も施さない．テキストモードではデータセット中のデータはEBCDICであるとみなされ，ネットワークから読まれるときにASCIIに変換される．

テキストとバイナリが区別されるため，ユーザがサーバに対しファイルの大きさを要求したときに，ファイルの大きさは要求されたモードによって異なったものになる．“テキスト”ファイルでは各レコードの末尾にキャリッジリターンが付加されるので，単純なバイナリモードで読んだときよりもファイルが長くなる．

こういった様々な問題を扱うためには，こちらがデータセットタイプや属性の処理について何を見たいと思っているかをメインフレームに知らせる手段が必要である．これは3つのレベルで可能である．

・サイトそれぞれのパラメータブロックで知らせる
・マウントポイント時に知らせる
・ファイルアクセス作成時に知らせる

サイトのデフォルトはサーバが初期化されるときに読み込まれるテキストファイルによって設定し，デフォルトの基本的な設定を定義する．ファイルの内容はだいたい次のようなものである．

```
mintimeout(1)
maxtimeout(1800)
timeout(90)
logout(1800)
# define primary and secondary allocations of space
# then define the number of blks(could do trks or cyls)
space(100,200),blks,blksize(6160),recfm(vb)
nfstasks(8)
```

マウント時にマウントコマンドの中で属性が指定される．

```
user% mount 'savant:sun.beepy,text,crlf,space(10,20)' /mnt
```

ファイルの指定には属性も含まれていることに注意せよ．属性はすべてカンマで区切られている．同じ属性の再指定はファイルをアクセスするときに行うこともできる．

user% vi '/mnt/file.txt, space(256, 256), recfm(vb), lf, dsorg(ps)'

このやり方から, Unix ではファイル名の中身に何が含まれていようともまったく関知せずにプログラムに渡すことがわかる. しかし, PC クライアントの場合は DOS がファイル名を解析しようとするのでこれができない.

これらのどのコマンドでも, ユーザは希望するファイルのタイプや, データの編成方法を指示している. ファイルアクセスの例ではユーザが SunOS の vi エディタをある MVS ファイルに対して使っているが, そこでファイルに割り当てる領域の量, 行末にラインフィードを使うことなど様々な形式を指定している.

もともとの MVS/NFS サーバはメインフレーム上の 4 種類のファイルを扱うことができる.

- 順編成
- 直接アクセス
- 区分編成
- VSAM

これらのファイルの中で, サーバは 3 種類のレコードフォーマットを扱える. 固定長レコード, 可変長レコード, 未定義 (バイナリ) である. またサーバはブロック化されたレコードやスパンレコードを扱える. 別個の RPC サービスでキーアクセスデータセットを取り扱うこともできる(ただし NFS の機能を介してではない).

今後の MVS/NFS のリリースには, MVS での Unix 式の名前と属性を完全にサポートした真の階層化ファイルシステムのサポートが含まれている. これによって MVS は Unix の NFS 環境での完全なファイルサーバとして振る舞うことができるようになる. MVS が定義するコロンブス以前の (つまり平面的な) モデルに, 完全な階層化ファイルシステムを追加することは, 近々刊行予定の Sun テクニカルレポートの主題である.

6.4.2 並行アクセス

ネットワークとメインフレームの両方からメインフレームのデータに並行にアクセスすることは MVS では問題である. データセットはクローズされるまで排他的なアクセスとなるのが(少なくとも排他的なアクセスが求められたときには)普通なのである. 問題は, 読出し/書込みのファイルシステムをマウントすると, MVS のサーバはファイルに対して排他的なオープンをしようとしてしまうということである.

しかし, NFS には内部状態がない. どうすればユーザが作業を終えたとわかるのか? NFS にはクローズという操作がないのである. クローズ操作がないことを (サーバはクライアントが動作を開始したり, 終了したりすることにいっさい関知しなくてよいということで) 強力な点であると考える人もいるが, MVS ではクローズは必要不可欠な操作である. データセットがクローズされると明示的なファイルのエンドマークが作られ, ファイルの内容が確実であることが保証さ

れる．ファイルエンドマークがない場合，データセットは壊れているとみなされてしまう可能性がある．

この問題は，タイムアウトしたときにクローズ操作を行うようにすることで解決している．例えばあるユーザが検索を行って，その結果データセットを割り当てる(オープンする)ことになったとしよう．このときタイマがセットされる．このタイマが時間切れになるとデータセットはクローズされる．つまりユーザが何回かデータを要求したときには，(特にタイムアウト時間が短い場合には）何回かオープンとクローズが行われるかも知れない．

6.4.3 VM/CMS NFS ファイルサーバ

VM オペレーションシステム上の NFS 実装は，属性とファイル名の対応づけに面白い考え方を示している．MVS と同じように，CMS ミニディスクをマウントするために，別個のマウントプロトコルが用いられている．これには他の変換サービスもすべて含まれている．

```
mount yktvmx:scanner.191
mount yktvmx:scanner.191,ro,pass = darkley
mount yktvmx:scanner.191,record = nl
```

最初のコマンドでは，ミニディスクがマウントされている．2 番目の例では，モードが読出し限定であり，パスワードも指定されている．3 番目の例では，レコードの末尾に改行を加えることで変換が指定されている．

デフォルトの変換方法は，何も行わないいわゆるバイナリモードである．つまりデータは単純にバイト列として読まれる．改行モードにすると改行コードが付けられる．さらにテキストモードでは ASCII から EBCDIC への変換も行われる．

名前の対応づけは少しやっかいである．ユーザが指定できる変換方式には 3 種類ある．

・大文字，小文字のどちらかにそろえる
・変換をせずに大文字，小文字の混在を許す
・ファイル名の変換を行う

ファイル名の変換の典型例は，CMS では許されていないような長いファイル名の場合である．また CMS は特殊文字や 2 つの部分からなるようなファイル名も好まない．図 6-16 に変換の例をいくつか示す．

Unixでの名前	CMSでの名前
good.data	GOOD DATA
bletch.	BLETCH #N
bletch	BLETCH #N2
free_for_all.c	FREE_FOR_ALL@C#N
%free_for_all.c	FREE_FOR_ALL@#N2

6-16 VMとNFSでのファイル名変換

6.5 ハードウェアによるNFSの実装

分散ファイルシステム，ディスクレスクライアント，そしてネットワークベースのアーカイブおよびバックアップサービスは，すべて企業規模のファイルサーバに対する強い要求が原動力になっている．特にこの要求に応えるためにAuspexやEpochなどのいくつかの専門会社ができている．

Auspexはネットワークファイルシステムをハードウェアを用いて解決する目的で特に設立された会社である．普通NFSサーバはVAXやSunなどの汎用コンピュータであるが，汎用のコンピュータはデータアクセス操作の頻度が高くなってくるとNFSサーバとして最適ではなくなる——ボトルネックが現れてデータの流れを阻害してしまうのである．Auspexの技術者はクライアントへのディスク記憶サービスだけを行うコンピュータアーキテクチャを開発し，これまでのサーバにあった様々なボトルネックを取り除いた．図6-17は代表的なサーバの構成を示したものである．

Auspexのアーキテクチャはメインプロセッサとインテリジェントな周辺装置で構成される．Auspexではこれを“機能並列処理”と呼んでいる．ホストプロセッサではUnixが走っており，これまでと同じようなセッション開始時の認証など管理的な作業を行っている．しかし，ファイ

階層	記憶種別	典型的な値
ワークステーション	主記憶	3～64メガバイト
	ディスクドライブ	0～500メガバイト
計算サーバ	主記憶	8～128メガバイト
	ディスクドライブ	250メガバイト～4ギガバイト
Auspex NS 5000	主記憶	34～114メガバイト(注を見よ)
	磁気ディスク	2～60ギガバイト
	8mmテープ	2.3ギガバイト
	4mmテープ	1.3～2ギガバイト
Epoch-1 InfiniteStorage サーバ	主記憶	8～20メガバイト
	磁気ディスク	2.28～7ギガバイト
	再書込み可能光ディスク	30～60ギガバイト
	WORM光ディスク	330～990ギガバイト
	8mmテープ	2.3ギガバイト/テープ
注: Auspexはマルチプロセッサ上に18メガバイトの固定された主記憶をもっており，このほかに、ユーザ構成可能な一時メモリデータキャッシュとして16～96メガバイトを追加することができる．		

6-17　記憶の階層

ルへの要求はほとんどがホストプロセッサを完全に飛び越して，特別構成のボードに直接送られるようになっている．

入ってくる要求は最初にイーサネットプロセッサが受け取る．最大4プロセッサの構成で計8個のイーサネットを別々に使うことができる．それぞれのプロセッサには完全なNFSプロトコル階層が組み込まれている．8個のイーサネットを使えるということは，単一のイーサネットLANのバンド幅を使い切ることなしに，非常に多くのクライアントに対応できるということである．

イーサネットプロセッサがパケットを受け取ると，パケットはファイルプロセッサへと渡される．この専用ボードは，クライアントから入ってきた要求のAuspexのファイルシステムへの対応づけを担当している．要求への対応はメインメモリキャッシュから行われることもあるし，記憶装置プロセッサまで行くこともある．

記憶装置プロセッサはSCSIベースのドライブのディスクアレイの管理を担当する．1つのドライブのラックは5つの1002メガバイトドライブである．1つのプロセッサには4つまでのラックを付けることができ，1台のマシンには3プロセッサまで付けられるので，計60ギガバイトの2次記憶とすることができる．

Auspexのアーキテクチャは，ディスクレスワークステーション(あるいはディスクスワップと少しの重要なオペレーティングシステムファイルだけのために，ごく小さなディスクしかもたない，いわゆる“データレスワークステーション”と呼ばれるもの）用として理想的である．Sun-4などのような典型的な汎用サーバの場合，10から20のクライアントに対応するのがせいぜいであるが，Auspexシステムではディスクレスにして50から100，データレスにして100から200のクライアントに対応できる．部局全体のような大構成では，ディスク供給にはコスト面で優れた方法であり，しかも汎用のサーバを解放して他の仕事に振り向けることができる．

6.5.1　無限容量の記憶装置：Epoch

AuspexはNFSプロトコルを大容量の2次記憶を使えるようにするための方法として使っている．もう1つの会社Epochは，同じプロトコルを違った用途に使っている．すなわち，企業ネットワークの中でのアーカイブおよびバックアップのための大容量の3次記憶にアクセスするためである．

Epochシステムは磁気ディスクと光ディスクを組み合わせて1つのファイルシステムのように見せてしまう――Epoch社のいう無限容量記憶装置(InfiniteStorage)という概念である．代表的なシステムは数種類の媒体を使う．磁気ディスク，8mmテープドライブ，再書込み可能光ディスク，単一書込み/複数回読出し(write once/read many : WORM)ドライブなどである．記憶能力は30ギガバイトからテラバイト(1000ギガバイト)の容量になるのが普通である．

クライアントはNFSプロトコルを使って，Epochサーバのファイルシステムをワークステー

ション上に“マウント”する．ワークステーションではファイルがローカルに存在するように見える．これを各ユーザにテラバイトのオンラインデータが与えられたと考えれば――CAD や地震解析には理想的である．

Epoch オペレーティングシステムはファイルがどの程度の頻度でアクセスされるかを監視している．ファイルがしばらくの間アクセスされない場合は，そのファイルを磁気ディスクから再書込み可能光ディスクに移す．そこでもファイルが使われないならば，テープか WORM 光ディスクに移す．クライアントから見ればファイルが利用可能なことは変わりがないが，だんだんと低速（かつ安価な）記憶媒体に移されていくのである．

考え方は，頻繁に使われるファイルを磁気ディスクなど高速な媒体に置くというものである．ファイルが使われていないならばもっと低速のシステムに置くことができる．その後ファイルがまたアクセスされたならば，磁気ディスクに戻して，速いアクセスができるようにする．すべてのファイルは依然として利用可能であるが，使われていないファイルは取り出すのに少しばかり時間がかかるようになる．

このようなある媒体から他の媒体へのファイルの移動はステージングと呼ばれる．1 日 1 回，一括して定期ステージングをするのが普通である．システムは磁気ディスクにあるファイルすべてを調べて，移動すべきかどうかを決める．

ファイルはある低水位線（low-water mark）に至るまでディスクから追い出される．この下限としては，使用領域 75%というのが管理者の 1 つの選択であろう．考えるべきはバランスの維持である．あまりたくさんのファイルが追い出されることになると，後になって戻さなければならなくなるだけである．ファイルが十分に追い出されないと，ディスクが後でいっぱいになったときに移動させなければならなくなる．

ときどき 1 日のうちでもドライブはある高水位線――例えば使用領域 90%――に至るかも知れない．こうなったときには Epoch システムはイベント駆動のステージングを使ってファイルを移動して空きを作る．ファイルのサイズと最後のアクセス時刻，そして管理者の意向により，低水位線になるまでファイルが移動される．

この無限容量記憶装置の概念はアーカイブのためにもバックアップのためにも役立つ．光ディスクにバックアップをとる場合によいのは，ユーザがファイルを復元しようとしたときにネットワークマネージャが間に入らなくてもよい点である．ネットワークマネージャはどのファイルがアーカイブ（オンラインに保存）されるべきか，そしてどのファイルがバックアップ（ファイルを復元するのにユーザの仲介が必要）されるべきかを決めることができる．

無限容量記憶装置の拡張としては，Renaissance と呼ばれる Epoch システムの新しいサービスがある．Renaissance は使われていないデータを動的にクライアントシステムから Epoch システムへと移動させることによって，ワークステーションを記憶階層に組み入れる．いったん Epoch システムに置かれれば，データは記憶の階層を下へとたどり，最終的にはテープか WORM ドライ

ブに置かれることになる.

6.5.2 新しい専門マーケット

Auspex と Epoch はどちらも同じプロトコル——NFS——を企業ネットワークの中の特殊な解決策を与えるために使っている. Auspex は大容量の 2 次記憶への超高速アクセスを提供し, Epoch は 3 次記憶の超大容量に特殊化している.

Auspex のような会社の代表的な顧客は, ソフトウェアデザインや CAD/CAM を行っている組織である. つまり, ワークステーションに大容量のファイルを必要とする環境である. このような環境では, いままでのサーバでは早々に埋まって, 飽和してしまう. Auspex の顧客である Adobe Systems 社や LSI Logic 社などの集積回路製造会社は, このようなサーバを高性能ワークステーションのデータ大容量化の要求に対応するために使っている. もちろん Adobe Systems 社や LSI Logic 社は Sun 社の大型顧客でもあり, Sun 社と Auspex 社の装置を組み合わせて統一ネットワークを形成している.

Epoch サーバの顧客の代表は石油会社や地質学研究者で, 非常に大容量の地震分析データを使える状態でもっていなければならない人たちである. 例えばカルフォルニア大学サンディエゴ分校では, 360 ギガバイトの Epoch システムを使っている.

より進んだ勉強のために

Chet Juszczak, "Improving the Performance and Correctness of an NFS Server," *Usenix Conference Proceedings,* Usenix Association, Berkeley, Ca., June 1990, pp. 53-63.

Bob Lyon and Russell Sandberg, "Breaking Through the NFS Performance Barrier," *Sun Tech Journal,* August 1989.

Rick Macklem, "Lessons Learned Tuning the 4.3BSD Reno Implementation of the NFS Protocol," *USENIX Conference Proceedings,* Usenix Association, Berkeley, Ca., January, 1991.

Bill Nowicki, "Transport Issues in the Network File System," *ACM SIGCOMM Computer Communication Review,* April, 1989.

Brian Pawlowski, Ron Hixon, Mark Stein, Joe Tumminaro, "Network Computing in the Unix and IBM Mainframe Environment", *Uniforum '89 Conference Proceedings,* March 1989.

R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, "Design and Implementation of the Sun Network Filesystem," *Usenix Conference Proceedings,* Usenix Association, Berkeley, Ca., Summer 1985.

Sun Microsystems, *Network Programming Guide,* Part Number: 800-3850-10, Revision A of 27 March, 1990.

———, *NFS: Network File System Protocol Specification,* RFC 1094.

———, *System & Network Administration,* Part Number: 800-3805-10, Revision A of 27 March, 1990. NFS, NIS その他の一般的な管理上の問題について述べられている.

Chapter 7

RPC Tool

7.1 コンパイラ

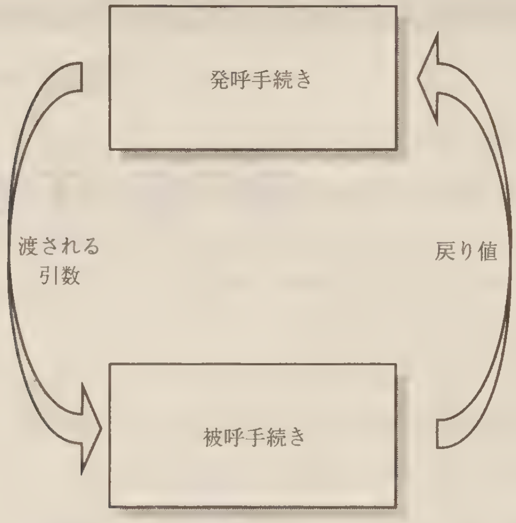
第5章では，遠隔手続き呼出し(remote procedure call : RPC)プロトコルによりいかにして手続きを他システムへ移動することが可能になるかについて議論した．RPC は，手続きを実行するために，リモートサーバに要求(request)を送ることができる糊づけ(glue)の役目を果たす．RPC サービスにアクセスするために，ユーザはRPC ライブラリに対する呼出しを直接書くこともできる．低水準の RPC 呼出しを書くことは，離れたコンピュータを透明なネットワークインターフェイスをもった現システムの拡張としてみることができるという，遠隔手続き呼出しの目的に反する．

RPC ライブラリを使用する直接的なアプローチでは，プログラマが外部のシステムをローカルシステムと異なるように扱う必要がある．コンピュータ内の呼出しの場合は単純な関数呼出しが実行される．離れたシステムの場合は RPC 要求が作られて，復号されなければならない．

RPC コンパイラによって，プログラマはすべての手続きがあたかも自ホスト上で行われているかのように扱うことを，かなりの程度まで可能にするような高水準言語を扱えばよくなる．手続きはまるで自コンピュータ内の手続きのように，パラメータを渡したり，いろいろなデータを返すように宣言される（図 7-1).

SunRPC プロトコルで利用できる RPC コンパイラには，もとからある rpcgen と，最近加えられた Netwise RPC Tool の2つがある．Rpcgen は無料でライセンスを得ることのできる RPC ソース配布において広く利用できるという利点があり，NFS 参照実装の一部をなしている．しかし，これは Netwise ソフトウェアより古い技術に基づいている．

Netwise RPC Tool は，RPC プロトコルをより使いやすくし，よりよい誤り検査，結合(binding)およびその他の高水準機能などの価値を加えた商用システムである．Netwise は入力として用いる仕様言語を定義している．この言語は単一コンピュータ内のプログラムの定義に用いられる C 言語の宣言と非常によく似ている．この仕様ファイルとアプリケーション内のいろいろな手続き



7-1 プロセス間通信

のそれぞれに対するコードは，コンパイラにより生成されるコードとシステムについてくるライブラリと一緒に束ねられる．

プログラマからみると，分散プログラムに対するネットワークインターフェイスは，非分散プログラムとのインターフェイスとよく似ている．種々のカスタム化(customization)機構を用いることで，この透明性を破って RPC 機構の制御を行うことも可能である．下位の(underlying)コンパイラがプログラムが分散配置しているときに生じる問題の多くの面倒をみるので，分散環境におけるプログラミングを透明にすることもできる．異なる計算機上の 2 つの手続きを分離することは様々の問題によって複雑である．

- 離れた手続きを位置づけて対応づけ（結合）を行うこと
- データを共通の形式で表現すること
- データを送受信すること
- 単一マシン環境では存在しないような誤りの取扱い

分散環境においては，単一マシンにおいて発生しないような誤りが起こる可能性がある．非分散アプリケーションでは単一動作ですむのと比べて，分散アプリケーションでは各々のマシン上でいくつものステップをふむ必要がしばしば生じる．分散環境では各マシンは独立に故障を起こす．1 つの構成要素の故障がトランザクションの一貫性(integrity)にどのような影響を及ぼすか，分散アプリケーションは判断できなければならない．

単一マシン上での手続き間の通信は，プロセス間通信(IPC)の特別の場合である．アプリケーションが分散されているときに，RPC は複数マシン環境に IPC 機構を拡張する．

7.2 Netwise RPC Tool

Netwise は, 1 つのマシンの上で走る一連の手続きを複数のマシンの上で走る一連の手続きに変換する RPC Tool というコンパイラを作っている. このコンパイラは遠隔手続きをどのように起動するか, そして複雑なデータ構造をどのように渡したり返したりするかの問題, そしてポインタの使用法や誤りの処理のような様々な問題を取り扱う. このコンパイラはプログラマが手続きを登録したり, 位置づけたり, 結合したりするための構文を提供している.

元の単一マシン用のアプリケーションは, 一連のユーザ手続きから構成されている. アプリケーションを分散化するアプリケーションプログラマはインターフェイス仕様を定義し, それが RPC Tool により処理されて 4 つの種類の手続きが追加される.

・クライアントとサーバのスタブ

・ディスパッチャ手続き

・パックおよびアンパック手続き

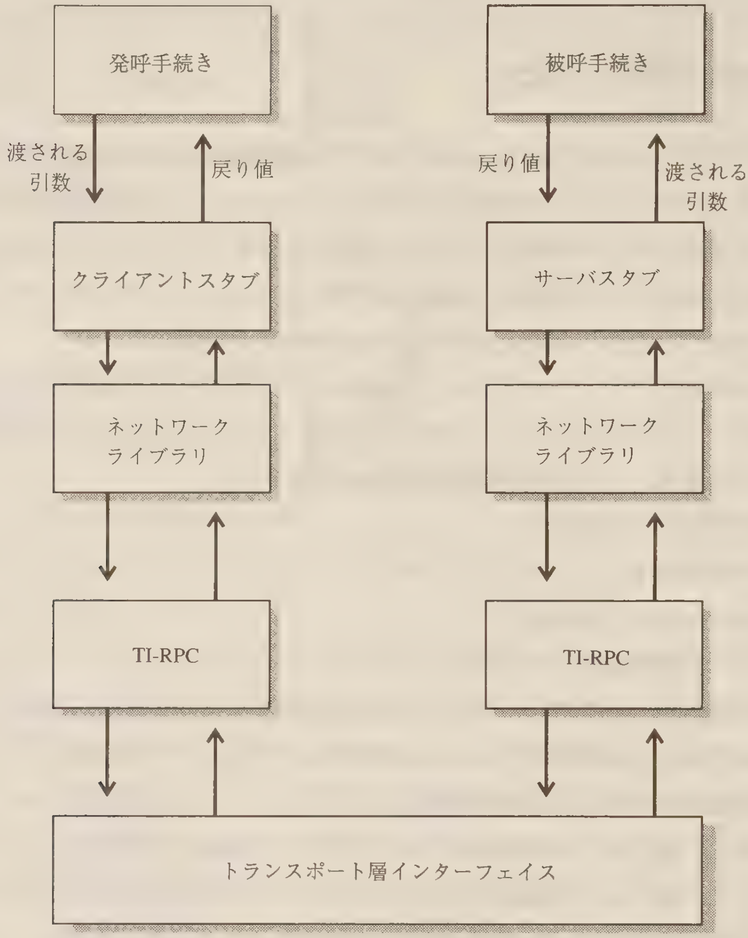
・ネットワークおよび支援ライブラリ

これらの付加手続きは元のアプリケーションの手続きとリンクされて, 分散アプリケーションの 2 つの構成要素, すなわちクライアントおよびサーバプログラムとなる.

プログラマの見地からすると, 重要な要素はネットワークライブラリとクライアントおよびサーバのスタブである (図 7-2). RPC Tool における多くの機構がデフォルトで動作するので (プログラマは単にどの手続きが分散配置されるかを示すリストを用意すればよい), プログラマが分散環境を完全に意識しないことも可能である. つまり, ネットワークは単なる 1 つの大きな分散マシンであるかのように取り扱われる. 複雑なアプリケーションに対して, プログラマはよりよい制御もしくは性能の向上のために RPC Tool 中の各要素を修正することができる.

クライアントおよびサーバスタブは, 単一マシンのプロセス間通信機構をエミュレートする. クライアントスタブは, クライアントアプリケーションに対してもとの被呼手続きと同じように見える. すなわち, クライアントスタブはもとの手続きと同じパラメータを受け取って, 同じ値を返す. パラメータを受け取った後, クライアントスタブは手続き呼出し要求をネットワークライブラリに伝える. ネットワークライブラリは, 引数を標準のネットワーク表現形成に変換し, 伝送に備えて要求を直列化することおよびリモートサーバに要求を配送することの責任を担っている. ネットワークライブラリは RPC Tool スタブ呼出しをトランスポート層呼出しにマップすることによってこれを実現する. それによってプログラムが特定のトランスポート層に依存しないようにすることができる.

RPC Tool により作成されたコードの透明性はとても重要な属性である. プログラマは, 各手続きの落ちつく先がどのサーバになるか気にかけずに, 手続きを設計する. ネットワークの下位



7-2 遠隔手続き呼出し

に位置するトランスポート機構もまたプログラマに対し透明である事実は，特に重要なことである．

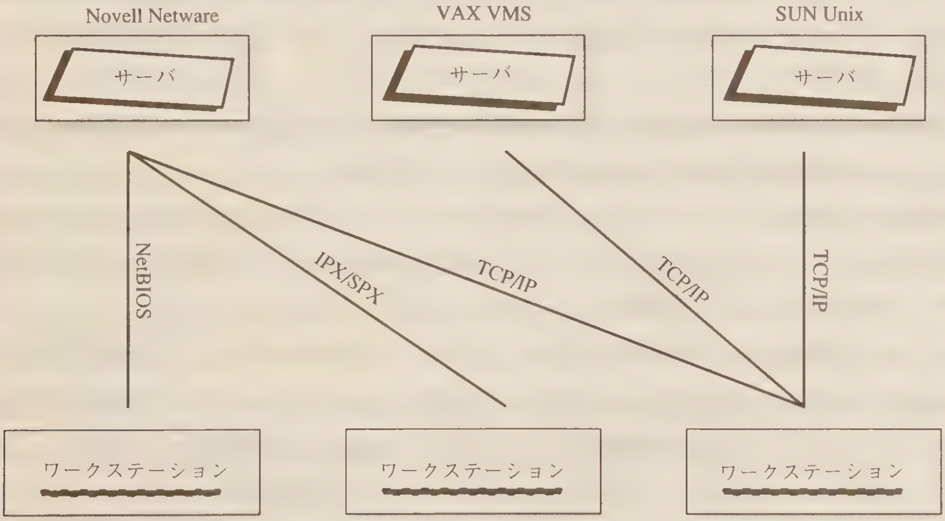
いったんデータが相手側マシンに配送されると，メッセージはサーバスタブに配達され，さらにサーバスタブがもとのユーザ手続きを呼び出す．クライアントおよびサーバスタブはもとのクライアントの本体を分散手続きに結合するネットワークの糊づけ役として機能する．サーバスタブは遠隔手続きを呼び出してしかるべき引数を渡す．その後サーバは戻り値を受け取ってメッセージに包み，ネットワークライブラリに伝える．ネットワークライブラリはクライアントにデータの結果を再び送り返す責任を担っている．そしてネットワークライブラリはクライアントスタブに応答(reply)を渡す．クライアントスタブは応答を受け取り，呼出し手続きが期待するような形にそれを包む．その後クライアントスタブは，あたかも自ホスト内のIPC機構が用いられたかのように，呼出し手続きに戻る．

ネットワークライブラリは，実行時に異なるトランスポート機構を選択することを許している(図7-3)．仕様ファイルをRPC Toolの異なるバージョンに与えることによって，異なるRPC機構をサポートすることさえも可能である．RPC仕様ファイルに書くことは，それが実行されるネットワークの種類を知ることなしにプログラムを書くことができることを意味している．

プログラムに標準的なC言語の構文が使用されているという仮定の下で，アプリケーションコードはマシン間ですばやく移動することができる．システムの資源を浪費しないようなプログラムをNovell Network Loadable ModuleとSunOS Unixサーバの両方に書くことができるだろう．Novellの世界におけるIPX/SPXやその他の世界におけるTCP/IPなどのようなトランスポート層の問題は，プログラマに完全に透明である．トランスポート層はRPCプロトコルで隠されており，そのRPCプロトコルはさらにRPCコンパイラとRPC Toolに隠されている．

RPC Toolが，移植性のよい(portable)分散アプリケーションを書くという問題に対する唯一のアプローチというわけではもちろんないが，この本の中で論じているような環境において，いくつかの大きな利点をもっていることも確かである．RPC ToolはNovellによりNovellのネットワーク上で用いられるRPC機構として採用された．RPC Toolの大きな魅力となっている点は，Netwiseが様々な他のコンピュータもサポートしていることである．Netwiseでサポートされているコンピュータには，80x86コンピュータすべて，VAX，Sun，IBM，その他多くのマシンがある．サポートされているオペレーティングシステムとしてはMS-DOS，Microsoft Windows，OS/2，DECのVMSとUltrixおよびUnixがある．

サポートされているネットワークの種類としては，Netware，NetBIOS，TCP/IP，DECnetおよびIBMのSNA/APPCがある．例えば，IBMのメインフレームはおそらくSNAネットワークプ



7-3 異種(Heterogeneous)環境

ロトコルを使用するであろう．Novell のワークステーションはSNA ゲートウェイと RPC Tool を用いて IBM 上のプログラムと通信することができる. 同様に, 3Com 3＋ワークステーションが NetBIOS シェルをもつ Novell ワークステーションと通信することを可能にするために NetBIOS を用いることができる.

7.3 手続きの宣言

RPC コンパイラはRPC 仕様を入力として受け取る．この仕様は一連の手続きの宣言である．プログラマは離れたマシンの上で走らせるアプリケーション内の各手続きごとに1つのエントリを用意する．次いで RPC 仕様は RPC コンパイラの入力として用いられる．コンパイラは仕様を受け取って，もとのプログラムとリンクされる一連の手続き，すなわちクライアントおよびサーバスタブと，パックおよびアンパック手続きを生成する．

マシン間で転送されるすべてのデータはパックおよびアンパックが施される必要がある．“データをパックする”とは，データをマシン独立の形式に翻訳することである．Netwise の場合，サポートされる形式は XDR と，ASN.1 BER として知られている ISO 標準である．“データをアンパックする” とはデータをマシン依存形式に戻すことである．メッセージが送られる直前あるいは受けられた直後に,クライアントおよびサーバスタブがパックおよびアンパック動作を呼び出す．

仕様は，基本的にリモートアクセスされる手続きのリストだけからなる非常に単純なものでよい．仕様の中に，誤りの回復のようなカスタム手続きを提供することも可能である．

プログラマがリモート手続きを定義するときに，その手続きが用いる引数と戻り値の定義を含める．プログラマは，ある変数がグローバルであり，明示的に受渡しすることなしに複数の異なる手続きからアクセス可能であると宣言することもできる．RPC Tool が大域変数をサポートしているのは，既存の多くのアプリケーションがこの手法をインターフェイスの一部として多用しているためである．RPC Tool において大域変数がサポートされていることにより，既存のアプリケーションを分散環境へ移植することが容易になっている．

データ構造の定義は，C プログラミング言語 (RPC コンパイラにより生成されたコードで用いられる言語である）と同じ方法を用いている．整数や文字のような単純なデータ型が定義でき，単純なデータ型を組み合わせて複雑なデータ構造を作ることができる．

ここで，データ構造の概念を説明するため，データベースのサーバとして動作する仮想的なプログラムを例としてとり上げよう．このサーバはクライアントからのデータ要求を受ける必要がある．プログラマは，いくつかの要素からなる DataRequest という構造体を定義する．最初の要素がコマンド型である．データベースサーバの場合は，これが操作がテーブルへのデータの読込み，更新，削除および追加のどれであるかを示す．この最初の要素はたぶん整数として定義されるであろう．構造体の 2 番目の部分は SQL 文(statement)そのものであり，文字列として定義され

る.

ほとんどのデータ型に対して，プログラマはRPC ToolにおいてもCプログラミング言語のANSI標準で用いられるのと同じ構文を用いる. 2つの例外が配列(array)と共用体(union)の複合データ型である.

配列は1つの単純なデータ型の集まりである．例えば文字の配列はテキストストリングを構成する個々の文字すべてを格納することができる．データベースは配列にうまく適合する．データベースのテーブルの各行は，配列のエントリと考えることができる．エントリが組み合わされて配列全体を構成する.

配列における問題点は，ある特定の要求に対して配列の中のすべてが埋められているわけではないことがしばしばあることである．配列は300列のデータをもつように設計されるかもしれないが, 特定の要求はそのうち12列しか取り出さないかもしれない．単一マシン環境においては，配列の中の使われていない空間は問題ではない(メモリが十分であるとして)．しかし，ネットワーク上では，空のセルを伝送することで1回の要求によって生成されるネットワークトラフィックの量に劇的な影響がある.

Netwiseでは，プログラマが配列の終端表現(terminal expression)を指定することを許し，事実上可変長配列を実現することによってこの問題に対処している．使われない最後のセルの後に，プログラマは次のセルに終端値（例えばヌル文字）を挿入する．パック手続きがデータをパックするときに終端値を認識して，パックを止める.

第2のデータ構造は共用体である. 共用体は複数の要素の集合からなり, ある時点ではそのうちの1つだけが有効となる. 単一マシン環境においては, 共用体の唯一の問題はメモリ空間の使用である. ネットワーク環境では, 共用体のメンバのうちの1つだけが使われているのにすべてのメンバをネットワークを介して送るのは，ネットワークの帯域幅とメモリとCPUサイクルの無駄使いである.

共用体の定義を補足するために, RPC仕様には共用体のどの変種が有効であるかを決めるための，実行時に実行される選択式が含まれている．有効な変種のみがネットワークを介してリモート手続きに送られる.

7.4 プロセスの結合

遠隔手続きと呼出しプログラムとの関係の確立方法は結合(binding)により指定される．2つのアプリケーションがコネクションを張りコマンドをやり取りする準備が完了したときに，結合は成立する.

もし手続きがめったに呼ばれないような場合には，永続的でない(nonpersistent style)結合が用いられる．永続的でない結合とは，遠隔手続きが呼ばれたときに下位のトランスポート機構上に

コネクションが張られることを意味している．値が戻ると，コネクションは切断される．ネットワーク上のコネクションは限りある資源であるので，永続的でない方式はその資源を節約するのに用いられる．めったに起こらない通信のためにコネクションを保つのでなく，コネクションが必要なときのみ張られる．

永続的でない結合は，各呼出しが別々のコネクションを張り，切断しなければならないことを意味している．コネクションの設定に伴う余分のオーバーヘッドのため，永続的でない結合は同じ呼出し者(caller)からよく呼ばれる遠隔手続きに対しては適当でない．

遠隔手続きへの呼出しを何回も繰り返すアプリケーションに対しては，いくつもの異なる呼出しがすべて同一コネクションを使用する永続的な(persistent)方式が用いられる．コネクションが設定され開いたままに保たれる．コネクションの識別子はメモリに保持されており，そのコネクションが後に続く呼出しの際に用いられる．アプリケーションがもはや相手側の資源の必要がないと決定すると，最後の手続き呼出しはコネクションを切断しようとしていることをサーバに伝える．コネクションのオープンとクローズは，通常プログラムの初期設定と終了手続きの中でのみ行われる．

これら両者の型の結合に対して，リモートサーバのアドレスは実行時に決めることができる．アドレスは名前およびインターネットワークのアドレスのいずれとしても指定することができる．サーバが走っているリモートコンピュータが常に同一であるなら，ターゲットを指定するのにインターネットワークアドレスを用いることができる．もちろん将来を予測することは難しい．また，アプリケーションがユーザに人気があり，増加する要求を処理するためにより大型のコンピュータの必要があるといった理由で，アプリケーションを異なるマシンに移すことは次第によく行われるようになる．

名前を使用することによって手続きの移動の柔軟性を増すことが可能となる．実行時に，DatabaseServer といったサーバの名前が指定され，ネットワークライブラリが適当な機構を用いて，その名前をインターネットワークアドレスに翻訳する．

例えば，Novell の環境においては，RPC ネットワークライブラリはサービスアドバタイズメントプロトコル (SAP) を用いる．NetBIOS の環境では，ネットワークライブラリは指定されたターゲットの位置を捜すために，NetBIOS の名前管理コマンドを用いるであろう．Sun の環境においては，NIS，NIS+，またはドメイン名前システム(Domain Name System)が用いられる．

サービスの抽象的な名前を用いることによって，ネットワークマネージャはクライアントおよびサーバ上のプログラムのいずれも変えることなしに，コンピュータ間でプログラムを移動することができる．サーバの位置が変わると，アプリケーションのサーバ制御プログラムの初期設定手続きが名前サービス (例えば NetWare ビンデリー(bindery)) にその新しい位置を知らせる．リモートサービスを要求しているワークステーションは，名前サービスからサービスの現在の位置を知らされる．名前づけの使用法は第 9 章で議論する．

7.5 クライアントスタブ

RPC Tool により生成されたコードがどのように動作するかを理解するためには，クライアントスタブから議論を始めるのがよい．クライアントスタブは外部にある遠隔手続きをエミュレートする，つまり渡された引数を受け取り，しかるべき戻り値を返す責任を担っている．

クライアントスタブは，状態機械として構成される一群のコードである．状態機械は，一連の状態を順に渡り歩くような手続きである．各状態において，状態機械はある一組の命令を実行し，その後適当な条件が満たされれば次の状態に進む．

クライアントスタブは最初，状態機械の開始(start)状態に入る．開始状態では，クライアントスタブで用いるデータ構造が初期化される．必要ならば開始状態でネットワークライブラリも初期化される．

次に，結合がネットワーク名に基づいていればマシンは検索(find)状態に入り，ネットワークライブラリの支援手続きを用いてサーバの名前をネットワークアドレスにマップする．ネットワークライブラリは，そのネットワークにおける翻訳のために必要とされる機構を用いる．

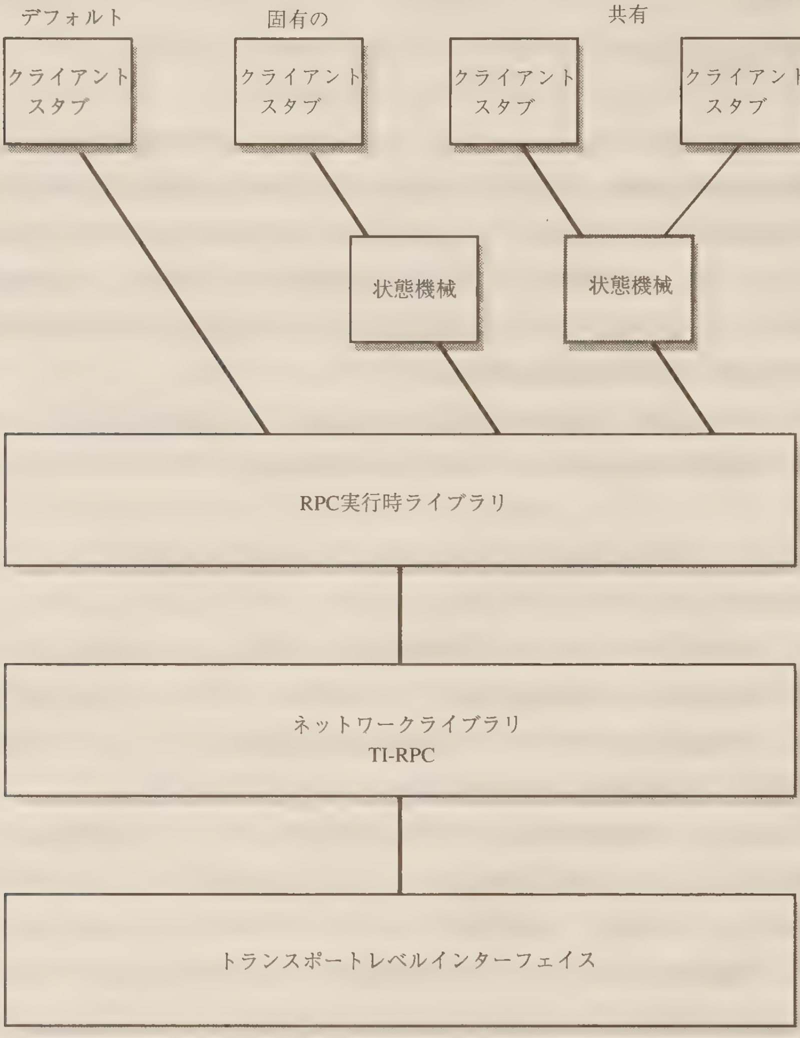
3番目に，マシンはコネクション状態に入る．ここではリモートサーバとのコネクションが確立され，コネクションの識別子がクライアント状態機械に提供される．もし結合が永続的な結合であるなら，コネクションの識別子はその後の呼出しで用いることができる．

4番目に，マシンはパック状態に入る．ここで，クライアントスタブの状態機械は，RPC コンパイラにより生成されたパック手続きを呼び出す．パック手続きはプロトコルデータ単位(PDU)として知られるメッセージを返す．PDU はメッセージシステムにおける封をされた封筒に相当し，トランスポート層に渡してネットワーク上を転送することのできるものである．

5番目に，マシンは送信状態に入り，PDU はネットワークライブラリに渡されて，ネットワークを介して送られる．クライアントマシンは，応答が受信されるのを待って受信状態に入る．応答が受信されると到着した PDU のヘッダをアンパックして，それが送り出した要求に対する応答であり，このソケットに迷い着いた漂流物でないことが確認される．その後，クライアントマシンは処理中に誤りが起こったかどうかを調べるために応答を調べる．もし誤りがなかったら，クライアントマシンはメッセージをアンパックして，しかるべきデータ構造に入れる．

最後に，永続的でない結合あるいは永続的な結合の最後の呼出しの場合には，状態機械はクローズ(close)状態に入り，遠隔手続きとの結合を終了させる．終了(finish)状態では，実行を止めて，制御を呼出し手続きに戻す．

これらの状態機械における多くの操作は，複数の手続きの間で共用し得る．パック手続きとアンパック手続きは異なっているものの，残りのコードの多くは再利用できる．RPC コンパイラは，状態機械を異なるクライアントスタブで共用することを許している．これにより，実行時に用い



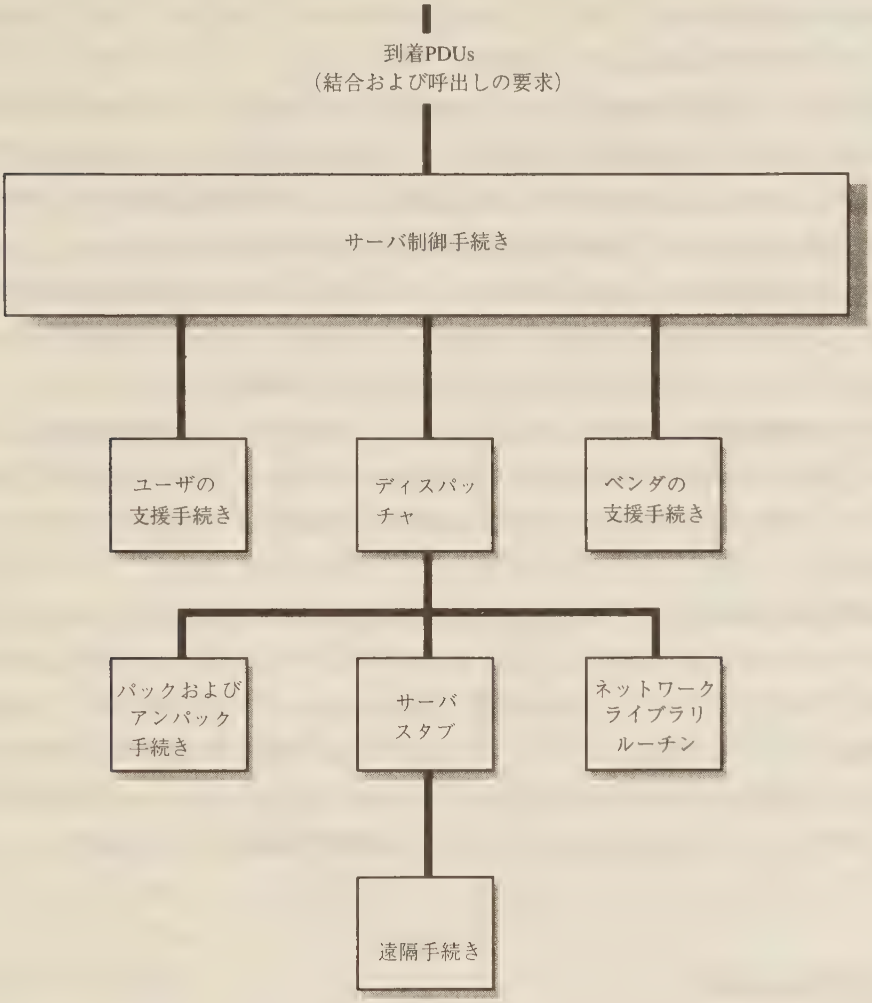
7-4　共有状態機械

られるメモリ量が減る．実際，もしプログラマが RPC Tool により提供されるデフォルトを用いて RPC 仕様のカスタム化をしなければ，ネットワークライブラリのサポートルーチンの中に組み込まれたデフォルトの状態機械が用いられる（図 7-4）．

7.6　サーバ制御手続き

サーバには，いろいろな遠隔手続きへのアクセスを調停するのに用いられるいくつかの構成要素がある．これらの構成要素はすべてサーバ制御手続きの制御のもとに動作する（図 7-5）．

サーバ制御手続きは，到着したメッセージを受け取って，それらがコネクションを初期化する



7-5　サーバ制御手続き

ための結合要求か実行すべき手続きであるかを判断する．もし到着した要求が実行すべき手続きならば，サーバ制御手続きはメッセージをディスパッチャ手続きに渡す．ディスパッチャはメッセージのヘッダをアンパックしてどの手続きが実行されるかを判断し，制御をしかるべきサーバスタブに渡す．サーバスタブは要求の本体をアンパックして，データをしかるべきデータ構造に入れる．その後サーバスタブは遠隔手続きを呼び出して，戻り値を待つ．

戻り値は応答メッセージに入れられて，ネットワークライブラリに渡され，ネットワークライブラリが応答をクライアントスタブに送る．クライアントスタブがさらに戻り値を呼出し手続きに返す．

RPC Tool は 3 つの型のサーバ制御手続きをもっている．それぞれ異なる型のサーバに適している．

・単一結合の制御手続きは，一度に1つの結合の要求を受け付けることができる．そして，要求を1つずつ実行する（単一タスク処理）．
・複数クライアント制御手続きは，いくつかの異なるコネクションを同時に管理することができる．この制御手続きは各々の到着メッセージを受け取って，どのディスパッチャを用いるかを決める．ただし，これはまだ単一タスクであり，1つの要求だけが一度にサービスされることに注意されたい．もし他の要求がくれば，待ち行列に入れられる．
・多重タスク制御手続きは，異なるコネクション上にいくつかの実行中の手続きをもつことができる．この制御手続きの多重タスク処理という性質は，多重タスク処理をサポートするオペレーティングシステムをもつコンピュータの上でしか動作しない．

多重タスク処理制御手続きに関連したものとして，多重スレッド(multithreaded)サーバ制御手続きがある．Unix は多重タスク処理を提供するオペレーティングシステムの一例である．Unix ではプログラマは多重タスク処理 SCP を使うであろう．OS/2 ではプログラマは多重スレッド SCP を使うであろう．

サーバ制御手続きはいくつかの段階を通過しなくてはならない．まず，ネットワークライブラリを用いてサーバの名前を登録する．次に，すべてのプロセスの結合を管理する責任をもっている．3番目に，到着したクライアントからの要求をルーティングするためにディスパッチャを呼び出す責任をもっている．最後に，サーバはすべてのメモリ資源を解放することやすべての処理中のコネクションを終了させることを含めて，サーバをシャットダウンすることも管理しなくてはならない．RPC コードの他の部分に関しては，サーバ制御手続きをカスタム化することが可能である．

7.7 カスタム手続き

RPC Tool 中で，プログラマがその動作をカスタム化することを許しているいくつかの機構がある．例えば，デバッグやより洗練された誤り回復手続きを提供するために，カスタム化を用いることができる．極端な場合，プログラマが状態機械の動作を変更したり，ネットワークライブラリを直接使用したりすることも可能である．RPC コードのカスタム化は4つの機構により行われる．

・“入口(entry)手続き”は状態機械に入るときに実行される．クライアントスタブの入口手続きは，その手続きが呼出し側のアプリケーションから呼ばれるたびに実行される．
・“出口(exit)手続き”は状態機械が終了するときに実行される．クライアントスタブの場合には，出口手続きは応答が呼出し側の手続きに返されるときに実行される．
・“フック”はある状態機械が異なる状態に入るときに実行される．例えば，マシン名がインターネットワークアドレスに翻訳された後(検索状態)，コネクションの設定を試みる前にフッ

クが活性化される.

・"トラップ"は例外条件が発生したときに実行される.これは通常誤りが発生したときである.

よく用いられるカスタム化は標準と異なる誤り処理である．例えば，プログラマはネットワークにより生じる特別の誤りを捜すようなトラップを設けることができる．プログラマは1回の誤りで直ちに呼出し手続きに戻るのではなく，再試行限度まで動作を再試行させることができる.

他の型の誤り処理は，アプリケーション特有の誤りを処理することである．メモリのある部分を見てどのような誤りが起こったかを判定するようなトラップを，サーバスタブの状態機械にもたせることができる．例えば，ユーザがデータを更新するために必要セキュリティレベルをもっていないという理由で，データベースサーバが更新を処理することを拒んだのかもしれない.

RPC Toolは，状態機械の中で何が起こっているかをプログラマが判断するのに役立つような一連のマクロ（ユーティリティプログラム）を提供している．例えば，ユーザはマシンの現在の状態を調べたり，ある状態から別の状態への流れを制御することさえ可能である.

カスタム化はサーバ上の制御手続き中のいくつかの場所で行える．まず，サーバ制御手続きはユーザ供給の初期化と終了手続きを常に実行する．例えば，ユーザは初期化の際にサーバの起動を記録することができる．サーバがシャットダウンするときには，ユーザ供給の終了手続きがそのことを記録し，サーバの動いていた時間を計算するであろう.

サーバループ手続きを用いて，より洗練されたサーバの制御を提供することも可能である．このサーバループ手続きは，到着した結合または遠隔手続きの要求をサーバが待っているときに実行される．ユーザ供給のサーバループ手続きを用いて，処理中のセッション数がサーバの容量に近づいてきているかを判断し，もし処理中のセッション数がサーバの容量に近づいてきていれば，他のサーバを（おそらく他のマシン上に）初期化することもできよう.

カスタム化の最も基礎的なレベルにおいては，ディスパッチャやスタブのような作り付けの機構の大部分を修正もしくはバイパスする．1つの可能性として，RPC呼出しの性質を同期から非同期に変えることが挙げられよう.

RPCモデルの下では，呼出し手続きが遠隔手続きを使おうとしたとき，呼出し手続きは結果が戻るまで処理を止める．これが同期処理，つまりあらかじめ決められた順序でものごとが起こるものであり，たいていのRPC機構はこのモデルに従う．もちろん多重スレッドの環境では，遠隔手続きの結果を待つ新たなスレッドを起動するので，主アプリケーションは処理を続けることができる.

非同期処理では，リモート要求が処理されているときに，呼出し手続きが動作し続けることが可能である．要求を送り出した後にすぐ戻るように，クライアント状態機械を修正すれば簡単な非同期モデルとなる.

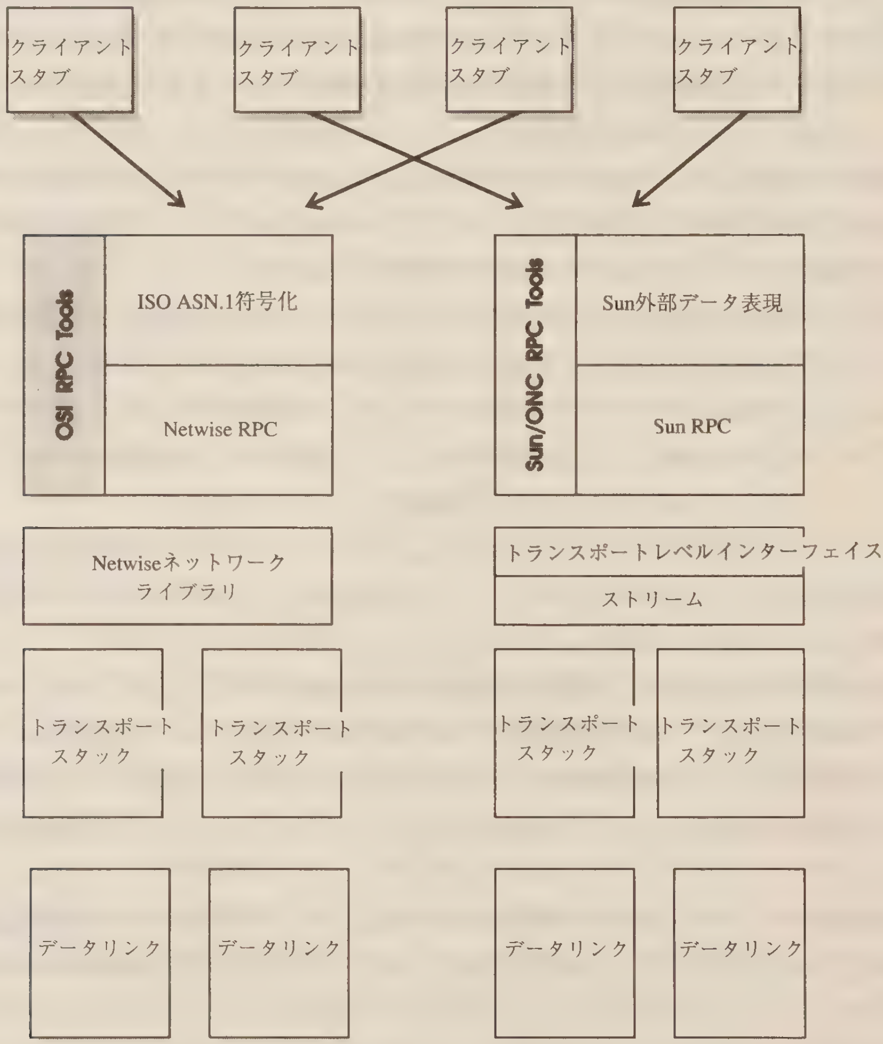
RPC Toolのようなものに対してこのような本質的な変更を行いたいと思うプログラマはあまりいないであろう．このような強烈な変更をするよりは，NeWSやX-ウィンドウのようなネット

ワークプログラミングの他の枠組みを調査すべきである.

7.8 ISO RPC 機構と SUN RPC 機構

Netwise RPC Tool のバージョン 2.0 は，抽象構文表記 1（ANS.1）基本符号化規則（BER）として知られる ISO 標準 8825 に基づいてデータを符号化する．その下に組み立てられるものはセション層のサポートを提供するための Netwise 機構である．例えばディスパッチャは到着したメッセージを受け取って，それをしかるべきサーバスタブの状態機械に転送する．

Netwise RPC Tool のバージョン 3.0 は，RPC 機構に関して Netwise モデルと Sun Open Net-



7-6　2種類のRPC機構

work Computing (ONC) モデルの両方をサポートする (図 7-6 参照). 機構の選択はコンパイル時に決められ, プログラマにとっては透明である. TI-RPC を用いれば, トランスポート層を実行時に選ぶことができる. TI-RPC では, プログラマが TCP と OSI に基づくプロトコルスタックの利点を使用でき, アプリケーションが実行時に適切な経路を選ぶことができる.

7.9 ネットワークへの他のインターフェイス

RPC Tool は分散プログラムを, クライアントとサーバ間のかなり洗練された相互動作を用いて開発する 1 つの方法である. しかし, 他のプログラミング手法により最もうまく取り扱うことのできるような型の動作もある. 特に, ネットワーク環境におけるユーザインターフェイスの制御は, X-ウィンドウシステムや Sun の Network Extensible Window System (NeWS) のようなグラフィックユーザインターフェイス(Graphical User Interface : GUI)標準により最もうまく取り扱うことができる.

X-ウィンドウシステムは MIT で開発され, Unix や DEC, IBM のワークステーション上で広く利用できる. X-ウィンドウシステムでは, 画面をもつコンピュータをアプリケーションプログラムから分離することが可能である. X-ウィンドウシステムは, ワークステーションの画面のビットモデルに基づいており, どのアプリケーションがワークステーションの画面のどの部分に書くことができるかを管理するための一連の規則を提供している.

X-ウィンドウシステムのような機構の利点は, 安いワークステーションを購入できることである. ワークステーションはまさに専用 X-ウィンドウ端末になり得る. これらの装置は, X-ウィンドウシステムのサーバのみを走らせ, アプリケーションを走らせることに関してはネットワーク上の他のサーバに依存する.

X-ウィンドウシステムは, 異なる種類のサーバがユーザのグラフィック画面, マウス, その他のデバイスと通信することを可能にしている広く利用可能な方法の 1 つである. これらのアプリケーションプログラムは, それらどうしで通信するために今度は RPC を用いることができる. 例えばメールシステムを考えてみよう. メールツールつまりユーザのグラフィックインターフェイスは, X-ウィンドウシステムのような機構を用いてユーザスクリーンと通信し, メールが到着するとメールボックスの旗を立てるような機能を実現するであろう.

これに対して, メールツールのプログラムはメーラやディレクトリサービスと通信するのに RPC を使用するであろう. また, メールを格納するファイルを他の場所に置くことができるように, おそらく NFS を用いるであろう.

X-ウィンドウシステムと RPC プロトコルによって, ネットワーク上に仕事を非常に広く分散させることが可能となる. 専用サーバを購入して, その構成をそれらが提供するサービスの種類に正確に合わせることができる. ディスクサーバは多量のディスクをもち, 非同期ポートはほと

んどもたず，プリントサーバは大きなスプール空間と多くのプリンタポートをもつように構成できる．

X-ウィンドウシステムは，ウィンドウの制御に関するよい標準であるが，たくさんのビットを描くことに基づくイメージモデルのようなものが欠けている．ウィンドウの中で動く言語を提供するために，Display PostScript が X-ウィンドウにしばしば追加される．Sun はこのアプローチを NeWS に採用し，DEC も同様に DEC ウィンドウで採用した．Phigs 3D グラフィックライブラリのような他のイメージモデルを使用することもできる．ウィンドウの上に組み立てられるのは，すべてのアプリケーションに対して共通の見かけと印象(look and feel)を与えるライブラリである．これによりメニューボタンのようなものが同じように働き，同じように見えるようになり，ユーザインターフェイスを用いることが容易となる．

OpenLook（Sun やその他多くのワークステーションで用いられている）や OSF で開発された Motif など，このような見かけと印象の標準は，ツールキットの形でプログラマが利用できることが多い．したがって，アプリケーションプログラマは2組のネットワークインターフェイスを見ることになる．すなわち RPC 仕様言語と GUI ツールキットである．さらに，プログラマはプログラムの作り方に関して的確な判断をするために，名前づけ，セキュリティ，ファイルアクセス（NFS）についてもおそらく知るべきであろう．

Sun のネットワークにおいては，ウィンドウ環境は OpenWindows として知られている．Open Windows は実際には X と NeWS の2つのウィンドウシステムである．X や NeWS に対する API のような様々なプログラミングインターフェイスやツールキットが利用可能である．NeWS は PostScript をもとに実現されているので，プログラマはアプリケーションを開発するにあたって，この描画言語の全機能を利用することができる．RPC Tool や TI-RPC と組み合わせれば，プログラマは洗練された分散アプリケーションを開発するためのツールをすべて手にすることができる．

より進んだ勉強のために

Sun Microsystems, *ONC RPC Application Toolkit for SunOS Installation and Operation Manual,* Part No: 800-4815-10, Revision A, April 1991.

———, *ONC RPC Application Toolkit Transport-Independent RPC Reference Manual,* Part No: 800-4813-10, Revision A, April 1991.

Chapter 8

ネットワークタイムプロトコル

8.1 なぜ厄介なのか?

Internet 上で他のホストに時報サービスを提供するホストの研究において，Delaware 大学の Devid Mills は明らかに正しい時刻表示を示すホストを合計 8455 個見つけ出すことができた．返された時刻の誤差は 11 ミリ秒から 11 日を越えるものまで様々であった．

なぜ時刻が重要なのか?　ネットワーク上では様々なサービスが同期のとれた時計に依存している．例えば，安全な(secure) RPC は確認証(verifier)を使うことによって，2 つのホストの会話中に他のホストが入ってこないようにしている．確認証は共有された DES キーで暗号化された時刻からなっている．

secure RPC を用いるホスト間では，最初のやり取りの際にウィンドウが決められる．ウィンドウというのは現在から未来方向のあるオフセット時間までの時間窓である．受け取った確認証に含まれる暗号化されたタイムスタンプがウィンドウの範囲外であった場合，その確認証は拒絶される．暗号化されたタイムスタンプとウィンドウによる制限を用いる目的は再試行を防ぐこと，つまり第 3 のホストが会話の進行を記録し，データを獲得するために後にその会話を再現することを防ぐことである．

2 つのホストの間で時計が同期していなければ，ウィンドウ機構は正しく動作しない．ウィンドウに収まらないタイムスタンプが受信されて，確認証は拒絶されてしまう．ホストは再びセッションを確立する過程を経なければならない（そして，これもまた時計が同期していないためにすぐに拒絶されるであろう）．

正しい時刻は分散データベースの更新が行われる環境においても重要である．2 つのリモートクライアントが共にデータを更新したりロックを獲得したい場合，通常，要求の時刻に基づいて更新の順序やロックの受け手が決定される．ここでも同期のとれた時計は基本となる．

ホストは正しい時刻を得るためにいくつかの方法を使用することができる．残念なことに，多くのコンピュータでは腕時計法が用いられている．つまり，システムプログラマが時計を見て，

その値を用いて時計を設定する．たとえ時刻が正しく設定されたとしても，時計が正確であり続けるとは限らない．多くの時計は電池でバックアップされたある種の装置によって維持されているが，それは1日に1秒まで狂う可能性がある（そしてめったに調整されることがない）.

われわれはすでに複数の時計間での2つの潜在的なエラー原因がわかっている．時計が異なった値に初期設定され得ることと，それらが時間の経過に伴って（異なる割合で）狂う可能性があることである．時刻が重要な同期手段になったならば，別々のホストの間の時計を合わせ，その状態を保つ方法を用意することは非常に重要となる.

TCP/IPプロトコル族においては，2つのかなり基礎的なプロトコルを用いることができる．ICMPタイムスタンプオプションがその1つである（もっともICMPタイムスタンプは日付を与えてはくれない）．2つ目の方法はRFC868にドキュメントされているTIMEプロトコルである．TIMEプロトコルはきわめて単純である．時刻サーバはUDPもしくはTCPの37番ポートを監視している．ポートが要求を受けると時刻を返す．時刻はUTC1900年1月1日の深夜0時からの秒数を示す32ビット数として送られる．ICMPタイムスタンプオプションはRFC781にドキュメントされており，基本的にはTIMEプロトコルと同じである.

8.2 NTP

さらに新しいプロトコルとして，一連のRFCと雑誌記事(この章の終わりの“より進んだ勉強のために”参照)にドキュメントされているネットワークタイムプロトコル(NTP)がある．NTPはICMPやTIMEよりはるかに洗練されたプロトコルであり，(非常に正確な時計を必要とする)高速ネットワークを特に考慮しつつ，大規模ネットワークにおける時計の調整を可能にする．NTPは以下のような2つの基本的な機能を提供する.

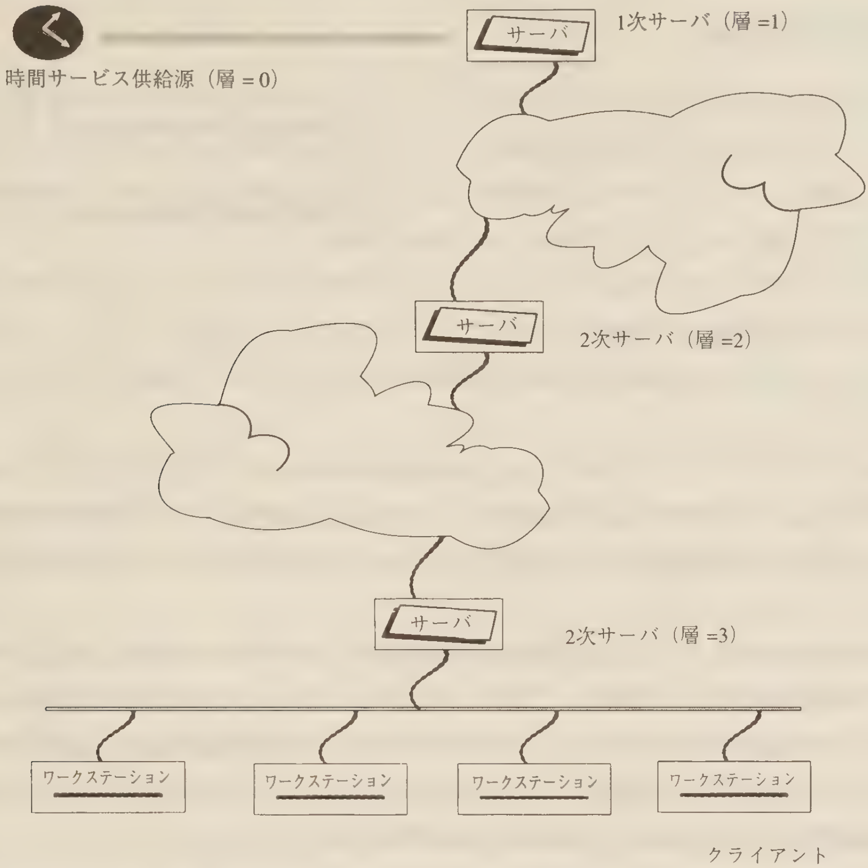
・サーバ群からクライアントへの時刻の分配を調整する方法

・時計の周波数を同期させる方法

周波数を同期させるというのは，ネットワーク上の時計を同じ周波数で動作させることである．これによりネットワーク上の時計は同じ速さで進むようになる．時刻を同期させるというのは，すべての時計を協定世界時（Coordinated Universal Time：UTC）に従った特定の時刻に一致させることである．時計を同期させるとは，周波数と時刻の両方とも一致させることである.

これを困難にしているのはInternet上の2つのコンピュータ間で生じる可変の遅延である．ローカルエリア環境においては，一連のいくつかの測定を通して時刻を数十ミリ秒の範囲内で同期させることはかなり容易である．広域環境での時計の調整は，何日にも及ぶ多くの測定を必要とする可能性がある.

時刻は最終的には1億分の数秒まで正確な国際原子時間（International Atomic Time：TAI）から得られる．地球の太陽を1周する時間の平均は(少なくとも現在は)ゆっくりと減少している



8-1 NTPネットワーク構成

ため，他の時刻形態として協定世界時 UTC があり，この時刻は現在 TAI と比較して 1 年に 1 秒にも満たないわずかな時間ずつ遅れ続けている――それゆえに暦年の 1990 年と 1991 年の間にうるう秒が導入されている．

UTC は第 1 の参照元として知られている．ネットワークタイムプロトコルにおいて，1 次サーバはこの第 1 の参照元に通信回線，無線通信または人工衛星により接続されている．さらに，時刻を他の 2 次サーバや 1 次サーバから取ってくる 2 次サーバがある（図 8-1 参照）．

1 次および 2 次サーバは階層的に並べられている．その階層を降りていくに従い，正確さのレベルが減少していくと考えられる．階層の各レベルには層(stratum)と呼ばれる数が与えられる．ルートとなる 1 次サーバの層は 1 である．時刻をルートから取得している 2 次サーバの層は 2 である．層は重要である．なぜなら，階層木を降りていくに従い，信頼性の低い時刻サーバである可能性が増すからである．

8.2.1 メッセージ交換

基本的なプロセスは，クライアントがNTPメッセージを1つ以上のサーバに送ることである．サーバは入ってくるデータグラムを受け取り，アドレスを入れ替え，メッセージのいくつかのフィールドに上書きし，チェックサムを再計算して，それを送り返す．各メッセージ交換につき4つのタイムスタンプがこのプロセスで得られる（図8-2参照）．

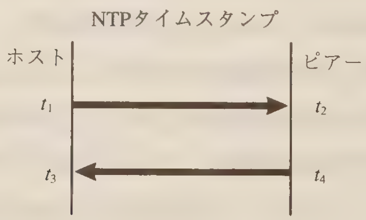
4つのタイムスタンプの組が与えられると，サーバは遅延とオフセットの2つの量を計算できる(図8-3および8-4参照)．遅延は1つのパケットがどれくらい長くネットワーク上にあったかを示し，これらのイベントを報告する時計が同期していないために，見積ることしかできない量である．

ローカルエリアネットワークにおいては，往復遅延は時計のオフセットのほんの一部に相当する量にすぎず，しばしば無視することができる．しかし，広域ネットワーク環境では遅延は容易にオフセットを越える可能性があり，遅延はかなり大きく変動することがあるのでさらにデータを処理することが必要となる．

どの時計のオフセットが正しいのかを決定する方法には様々なものがあり，異なるサーバからいくつかの情報を集めることに基づいて行われる．あるピアーに対する往復遅延時間と時計のオフセットは，4つの最新のタイムスタンプから計算される．時間の経過と共に，遅延とオフセットの見積りの組が蓄積される．問題はこれらの数の組が与えられたとき真のオフセットと遅延（または真の値の比較的正確な見積り）を決定する必要がある．

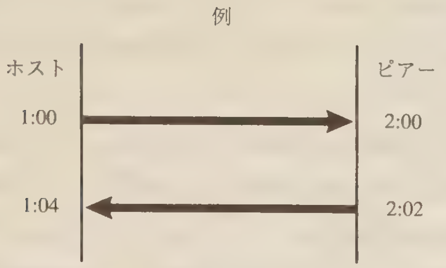
見積りの源がかなり離れていたりネットワークが混んできたりすると，遅延は通常増加する．遅延が増加するに従い時計のオフセットもまた増加することが研究により示されている．これは道理にかなう．なぜならパケットはより長い時間ネットワーク上にあり，したがって時計のオフセット数は強制的に増やされるからである．

はぐれ者を取り除くために，NTPは到着する見積りをフィルタを用いて遅延の増加順に並べ替える．遅延が少ないものほどよい見積りと考えられる．再び，8つのスロットをもつレジスタが用いられる．そのレジスタはオフセットと遅延のサンプル，そしてNTPバージョン3ではさらに分



$$\text{オフセット} = \frac{(t_4 - t_3) + (t_2 - t_1)}{2}$$

8-2 NTPタイムスタンプ



$$オフセット = \frac{(-58)+(-60)}{2}$$

$$遅延 = (t_4 - t_1) + (t_3 - t_2)$$

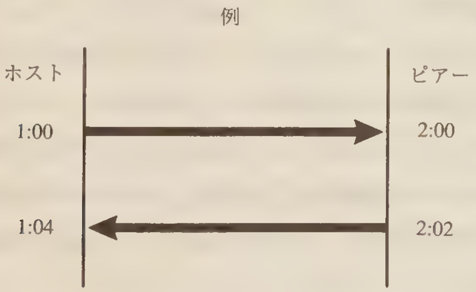
8-3 オフセットの計算

散の見積りを保持している．まず最初に最初の 8 つの見積りの組をとり，それらをレジスタに加える．新たな見積りが入ってくると，それはレジスタの最も右の位置に加えられ，最も左のものを落とす．

あるピアーに対する遅延とオフセットの見積りの組が得られたら，次の問題はどのようにしてどのピアーが正しいか決定するかである．一般法則として，大多数の時間供給源は正しく，“間違った時計”の数は比較的少ないと仮定することができる．もう 1 つ可能な仮定は，大多数の正しい時間供給源は真の UTC にきわめて近い時間を提示するということである．

われわれはさらに 2 つの仮定をすることができる．まず最初に，層の数が小さい供給源，つまりルートに最も近いものは正しい時間の見積りを提供する可能性が比較的高いということが仮定できる．最後の仮定は，分散の小さな，つまり見積りのばらつきが少ない供給源はより正確である傾向があることである．

ここでの問題は，それぞれが遅延とオフセットの見積りとそれらの見積りの分散をもつピアーの集合が与えられたとき，どれをその計算機上の時計を更新するのに用いられる真の源と考えるかということである．NTP はアクティブなピアーのリストをとり，まずそれを層により並べ替え，



$$遅延 = (4) - (2) = 2$$

8-4 遅延の計算

さらに分散により並べ替える．ピアーをリストに含める前に，そのピアーはまずいくつかの基本的な健全性チェックを通る必要がある．例えば，そのピアーに対する時間の源はそのホスト自体であってはならない．さもなければ同期ループができてしまう．

他にもいくつかの健全性チェックが用いられる．分散を示す数は半分は埋まっているサンプルレジスタに基づいている必要がある．さもなければその分散は非常に低いサンプリングレートに基づいたものとなる．これにより壊れた（または非常に新しい）アソシエーションからのデータの使用を防ぐことができる．1つのピアーも健全性チェックを通らない場合，その計算機上の時計は自然のペースで動き続けるだけである．

一度そのリストが正当性チェックで切り詰められると，それは最大サイズ(現在5)に設定される．さらに，そのリストは層数が2以上になるとすぐに再び切り詰められる．これは候補リストが2つ以上の異なるレベルの時間階層を決して含まないことを意味する．

最後に，そのリストは再び並べ替えられる．今回はそれはまず層により，そしてさらにそのホストとピアー間の距離により並べ替えられる．それから残っているピアーのうち最大分散をもつピアーが捨てられる．この手続きが繰り返され1つのピアーが残る．そのピアーが時計の源になる．

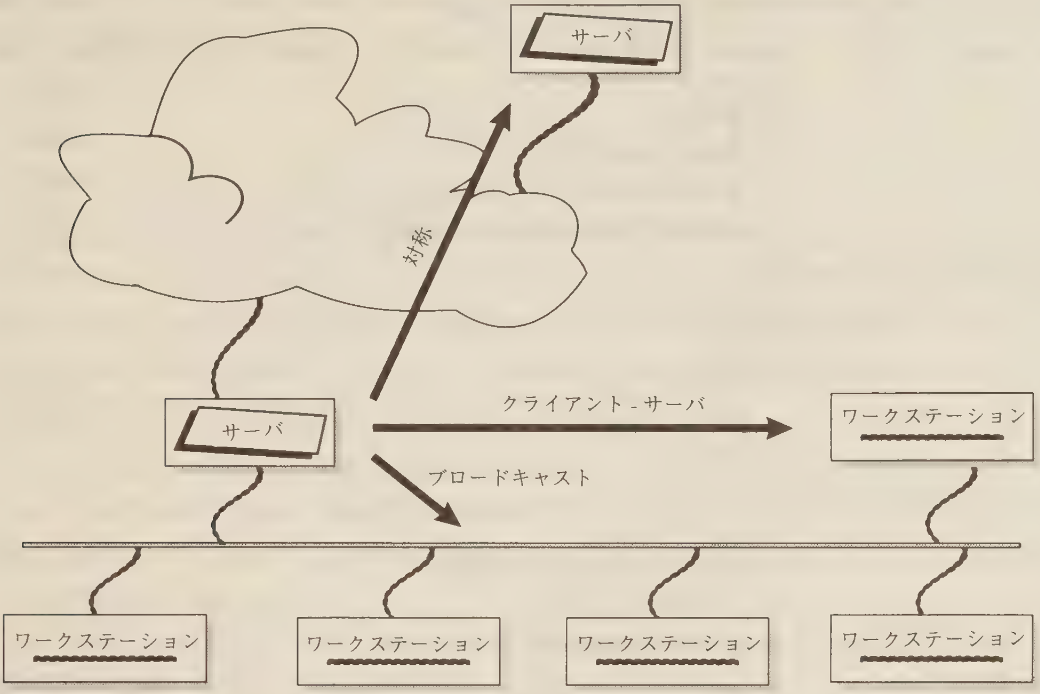
8.3 アソシエーション

2つの計算機間のアソシエーションを考えた場合，計算機は5つのモードのうちの1つにある(図8-5参照)．

・対称的にアクティブ
・対称的にパッシブ
・クライアント
・サーバ
・ブロードキャスト

2つの対称モードは2つのピアーがお互いの参照源になり得るような場合に用いられる．これら2つのモードはあるホストが，ピアーを同期させること，およびピアーにより同期されることを望んでいることを示している．アクティブモードは階層の下の方にあって頻繁にメッセージを交換する2つのピアー用のモードである．パッシブモードは階層の上の方にあって，つまり小さな層番号で，あまりメッセージ交換をすることのないノード用のモードである．パッシブなピアーは要求を待つが，アクティブなピアーはあらかじめ構成されたテーブルを使ってパッシブなピアー，さらには1次ルートを指し示すことができる．

ブロードキャストモードは多くのワークステーションがつながった高速LAN用のモードである．ブロードキャストモードでは，1つ以上のサーバが定期的にメッセージをブロードキャストし



8-5 NTPの相関図

ている．ワークステーションは蓄えられた遅延時間の数値を用いて実際の時間を決定する．

クライアント・サーバモードはブロードキャストモードより高い精度を必要とする環境用のモードである．クライアントは定期的にサーバにメッセージを送り，同期を行うためにサーバから返ってくる情報を受け入れる．クライアントのみが同期を受け入れ，サーバのみが同期を与えることができる．

8.4 NTP プロトコル

NTP はユーザデータグラムプロトコル (UDP) 上に構築されている．TCP ではなく UDP を選んだのは，データグラムプロトコルは失われたパケットの再送を試みないためである．NTP パケットは紛失してもかまわないものなので，一定の時間後に再送するよりも落としてしまった方がよい．すべての NTP メッセージは基本的に同じであり，ヘッダといくつかのタイムスタンプからなる（図 8-6 参照）．

NTP プロトコルでは，アクションはイベントに反応して起こる．典型的なイベントは特定のピアーに対するタイマが時間切れになった場合である．イベントの他の発生例はメッセージをピアーから受けた場合，操作コマンドが入力された場合，システムフォールトが起こった場合（例：

フィールド	長さ	説明
うるう秒指標	2ビット	本日の終わりに差し迫ったうるう秒が挿入もしくは削除されることを示す警告．値を以下に示す．
		00：警告なし
		01：最後の分は61秒である
		10：最後の分は60秒（訳注：59秒が正しい）である
		11：警告状態：自計算機内クロックが同期していない．
バージョン数	3ビット	現在3
モード	8ビット	現在の動作モード
		0：未定
		1：対称的にアクティブ
		2：対称的にパッシブ
		3：クライアント
		4：サーバ
		5：ブロードキャスト
		6：ntp制御メッセージのために保留
		7：個人使用のために保留
層	8ビット	1が1次参照，2-255が2次サーバ．層0は未定
状態監視間隔	8ビット	ホストからピアーへ送られるメッセージの間隔．ホストは常にそれ自身の状態監視間隔とピアーの状態監視間隔の最小値を用いる．このフィールドは2の乗数を秒として示した値である（64秒の最小間隔は6となる）．
精度	8ビット	自ホストの時計の精度を2の累乗に最も近い秒で表したもの．
同期距離	32ビット	1次参照源からの往復遅延の見積りをビット15とビット16の間に小数点をはさんで秒で表したもの．
同期分散	32ビット	1次参照源へのサンプルの分散の見積りをビット15とビット16の間に小数点をはさんで表したもの．
参照時計識別子	32ビット	層0と1に用いられる．
	NIST	NIST公衆モデム．
	ATOM	原始時計．
	コールサイン	一般的な無線源．
参照タイムスタンプ	64ビット	同一計算機上の時計が最後に更新された時間．
発生タイムスタンプ	64ビット	最後に受け取ったNTPメッセージが発生した（到着時に送信タイムスタンプからコピーされた）時間．
受信タイムスタンプ	64ビット	最後のメッセージが受信された時間．
送信タイムスタンプ	64ビット	最後のメッセージが送信された時間．
認証	96ビット	オプション．鍵識別子とメッセージの中身の暗号化されたチェックサム．

8-6　NTPヘッダ

時計が動作しない)である.イベントが起こった際に起動される手続きは基本的に以下の 3 つである.

- 送信
- 受信
- 更新

送信手続きはタイマがピアーに対して時間切れを起こしたときに呼ばれる.タイマはサーバモード（クライアントがクライアントサーバモードのトランザクションを初期化する）を除くすべてのモード中のすべてのピアーに対して保持されている.タイマが時間切れを起こすと, ntp メッセージが送られタイマはリセットされる.タイマをリセットするのに使われる数は,そのピアーへ至る経路のネットワーク状態に基づいている.

ntp メッセージが到着すると受信手続きが呼ばれる.メッセージがすでに存在するピアー対ピアーのつながりに一致しない場合,新たなつながりが作られる.いくつかの正常性チェックをした後,到着メッセージはメッセージの往復遅延とこのホストに関係のあるピアーの時計のオフセットを計算するのに用いられる.

ピアー対ピアー環境において,メッセージ交換は頻繁に行ったり来たりすることで行われる.メッセージ送信,受信のたびにメッセージにはタイムスタンプが付けられる.メッセージを続けて見ることにより,往復遅延を計算することができる.

オフセットもまた,頻繁に交換されるタイムスタンプに基づいている.あるパケットのペア(ホストからピアーに行くものと,応答として返ってくるもの）に対し,時計のオフセットは以下のように概算される.

- このホストがメッセージを送った時間とそれがピアーにより受信された時間の間に経過した時間の量を測定する.ただしここでは 2 つの異なる時計を見ていることに注意すること.これは時計のオフセットの 1 つの測定値である.
- ピアーがメッセージを送った時間とホストがそのメッセージを受信した時間の間に経過した時間を測定する.これは時計のオフセットのもう 1 つの測定値である.
- これら 2 つの値の平均をとる.

時計のオフセット値は 2 つの時計間のずれの最大値の見積りと比較することで正常性がチェックされる.ただし,メッセージ交換の進行に伴って往復遅延とオフセットは継続的に更新されることに注意すること.

3 番目の手続きは新たな遅延とオフセットの見積りに基づいた更新手続きである.しかし,その計算機上の時計が更新される前に, 1 つ以上のピアーからの結果に基づく重みの付けられた投票手続きが行われる.これが行われるのは,以前述べたサンプルと分散を測定する過程においてである.

8.5　NTP 制御メッセージ

NTP システム提供者に対する大部分の制御は，簡易ネットワーク管理プロトコル(SNMP)と MIB データベースの拡張を用いて行われる．NTP と SNMP サービスは異なる比率で実装されているので，SNMP には代わりが存在する．これが NTP 制御メッセージである．制御フォーマットには再結合のための領域があるが，これは IP ホストが 576 オクテットよりも大きなパケットを再結合する必要がないこと，および管理メッセージがそれよりも大きくてもよいようにするためである．図 8-7 と 8-8 に NTP 制御メッセージのフォーマットを示す．図 8-7 に一連のステータスワードの存在を指し示す制御メッセージヘッダを示す．図 8-8 にステータスワードを示す．

8.5.1　不良サーバの扱い

経験則によれば常に何かが壊れている．NTP に関していうならば，これは定められたサーバもしくはクライアントが壊れている可能性があること，そしてプロトコルがこの問題を取り扱うこ

フィールド	長さ	説明
バージョン数	3ビット	現在＝3
モード	3ビット	値6はNTP制御メッセージを示す．
応答ビット	1ビット	命令の場合0，応答の場合1．
エラービット		通常の応答の場合0，エラー応答の場合1．
継続ビット	1ビット	最後の断片の場合0．
操作コード	5ビット	0：保留．
		1：ステータスを読み込む 命令／応答．
		2：変数を読み込む 命令／応答．
		3：変数に書き出す 命令／応答．
		4：時計変数を読み込む 命令／応答．
		5：時計変数に書き出す 命令／応答．
		6：トラップを設定する アドレス／ポート命令／応答．
		7：トラップの応答．
		8-31：保留．
順番	16ビット	命令と応答を一致させるのに用いられる．
ステータス	16ビット	システム，ピアー，または時計の現在のステータス
つながりID	16ビット	つながりを識別する整数
オフセット	16ビット	データ領域の最初のオクテットのオフセット
カウント	16ビット	データ領域の長さ
データ		468オクテットまでのメッセージデータ
認証指標	96ビット	オプション

8-7　NTP制御メッセージ

フィールド	長さ	説明
システムステータスワード		つながり識別子は0
うるう秒識別子	2ビット	00：警告なし.
		01：最後の分は61秒である.
		10：最後の分は59秒である.
		11：警告状態
時計の源	6ビット	0：未定
		1：目盛のついた原子時計
		2：VLFまたはLF無線
		3：HF無線
		4：UHF無線
		5：ローカルネット
		6：UDP/NTP
		7：UDP/TIME
		8：眼球と腕時計
		9：電話モデム
システムイベントカウンタ	4ビット	システムステータスワードが最後に送られたときからのシステム例外イベントの数.
システムイベントコード	4ビット	最新の例外イベント.
		0：未定
		1：システムの再立ち上げ
		2：システムまたはハードウェアフォールト
		3：新しいステータスワード（うるう秒ビットまたは同期の変更）
		4：新しい同期源または層
		5：時計のリセット
		6：不当な時間または日付け
		7：時計例外条件
ピアーステータスワード		命令に対する応答において
ピアーステータス	6ビット	各ビットがフラグとなるピアーのステータス.
		0：構成済み
		1：認証が起動されている
		2：認証確認済み
		3：到着可能性確認済み
ピアー選択	3ビット	提供される時間に基づくピアーのステータス
		0：拒絶
		1：正当性チェックを通った
		2：正確性チェックを通った
		3：切り詰めチェックを通った

8-8 NTPステータスワード

フィールド	長さ	説明
		4：部外者チェックを通った
		5：現在の同期選択
		6：現在の同期源
ピアーイベントカウンタ	4ビット	最後のピアーステータスワードが返されてからのイベントの数．
ピアーイベントコード	4ビット	最後のピアー例外イベント．
		0：未定
		1：ピアーIPエラー
		3：ピアー到着不可能
		4：ピアー到着可能
		5：ピアー時計例外
クロックステータスワード		つながりIDにより示されるホストのシステム時計またはピアー時計を示す．
ピアーステータス	8ビット	0：予定通りに動作する時計
		1：応答タイムアウト
		2：間違った応答フォーマット
		3：ハードウェアまたはソフトウェアフォールト
		4：伝搬の失敗
		5：間違った日付けフォーマットまたは値
		6：間違った時間フォーマットまたは値
時計イベントコード	8ビット	最新の時計例外イベントを識別する．

8-8（続き）　NTPステータスワード

とができる必要があることを意味する．これらの問題の処理を助けるためにいくつかの手法がプロトコル中に組み込まれる．

・到達可能性の決定
・認証
・順番チェック
・状態監視速度の調整

NTPはコネクションのないUDP上に構築されているが，ピアーに関連する状態情報を維持することは意味がある．到達可能性の決定は，単にシステムが送信メッセージに対する返事を受けているかどうかの追跡を行うことによって実現されている．正常な場合，すべてのメッセージは返事を受け取る．

2つの送信の間に1つの返事も受け取れない場合，8ビットレジスタの1つのビットが0にセットされ，ビットが左にシフトされる．2つの継続する送信があるごとにその手続きが繰り返される．レジスタがすべて0の場合，8回連続して送信があったことになりピアーは到着不可能とみな

される．

認証は NTP のオプションである．リモートユーザに自分の計算機上の時計の更新を許すことは，権利のない，悪意のあるユーザの場合問題となる可能性がある．Internet アドレスを特定の範囲と適合させせることに基づいた単純なアクセス制御手法が NTP で用いられている．これは簡単なアクセス制御手段であり，認証ではない．

非常に安全を必要とする環境では，オプションで暗号化認証手続きを用いることができる．DES 暗号化とチェックサムを用いることで，メッセージが正しい源からのものであり，変更されたことがないことを保証することができる．

順序チェックは頑丈性の確保を助ける第 3 の機構である．この機構はメッセージ内のタイムスタンプの一連の正常性チェックである．例えば，受信メッセージの送信タイムスタンプが前に受け取った送信タイムスタンプと同じである場合，メッセージは重複していると想定することができる．さらなるチェックによりタイムスタンプが単調増加していることを確かめることで，パケットの順序乱れを防ぐことができる．

状態監視速度に対する細心の制御は頑丈性を確保するために用いられる第 4 の機構である．1 次時刻サーバは 300 以上のピアーをもつ可能性がある．過剰な負荷を防ぐため，状態監視間隔をかなり長くとることが望ましい．最初は，特定のアソシエーションに対するポーリング間隔は時計を同期させるため非常に短い．ポーリング間隔はつながりの始めでは 1 分程度の短いものであるが，時計がきちんと同期するに従い 17 分まで長くなる．

8.6 Internet 上での実装

Internet 上ではカナダ，ドイツ，ノルウェー，イギリスそしてアメリカにおいて一連の計算機が 1 次時刻サーバとして動いている．1 次サーバは直接国の時刻サービスにつながる．1991 年 5 月には，少なくとも 58 のそのような 1 次サーバがあった．これらの計算機の多くは直接国のバックボーンネットワークにつながっており，すべての利用者から利用可能となっていた．残りは特定地域のネットワーク上にあり，特定地域のネットワーク環境の範囲内で用いることを意図したものであった．

1 次サーバは継続的に互いにメッセージを交換し合い，高いレベルの冗長性を提供している．ある 1 次サーバ上のタイムコーダが動かなくなった場合，そのサーバはピアーの 1 つを情報源として使うことができる．1 次サーバはかくして 2 次サーバとなり，利用できる最も低い層(この場合 1) と最も小さな同期距離をもつ 1 次サーバに接続する．

Internet 上の 2 次タイムサーバは数千の計算機からなる．より頑丈な構成は一組のサーバが互いにバックアップを提供し合うというものである．Illinois 大学と Delaware 大学は互いにそのようなアクティブなバックアップシステムを提供している．

8.6.1 ちゃんと動くのか？

Internet におけるタイムサーバの正確さを確かめるために，David Mills は 3 つの時間要求プロトコル(ICMP, TIME, NTP)のそれぞれを，彼が見つけることができたすべての Internet アドレスに対して使用した．SRI International で集められたリストを用いて，全部で 112 370 個のホストが見つかった．ここで簡単に説明する結果は RFC 1128 の記事に細かく述べられており，後に更新され *Computer Communications Review* の記事になっている（“より進んだ勉強のために”参照のこと）．

ファイル中の各エントリに対し，実験では 3 つのプロトコルの各々を各ホストに対して使用することを試みている．1 秒以内に返事を受け取れない場合，プログラムは再試行を行う．さらに 1 秒後に返事を受け取れない場合，その試行を断念する．ICMP のホストまたはネットワーク到達不可能メッセージが受信された場合にもその試行を断念する．

プログラムは NIST 時計の数ミリ秒以内で同期したホスト上を走る．3 つのタイムプロトコルの各々に対して，8 個までの返事に対するオフセットが各ホストで蓄積される．最大，最小，そして平均がさらにデータファイルに挿入される．

最終的に観測された 107 799 個のホストのうち，94 260 個がある種のエントリ（つまりエラーメッセージまたはデータ）となった．この全ホスト数のうち，22%（20 758 個）が正当である可能性のあるタイムスタンプを返した．

プロトコルは特定のホストに対して 1 つだけが使われるという条件でランクづけされた．NTP が第 1 位で，TIME が 2 位，ICMP が 3 位となった．さらに切り詰めた後でリストは 8455 ホストに削減された．3694 個が ICMP タイムスタンプを提供し，7666 個が TIME プロトコルに応答，そして 789 個が NTP の要求に応答した．789 個のホストに加え，NTP 同期サブネットを監視プログラムを用いて探索することにより，さらにいくつかの NTP ホストが加わり合計 946 個になった．

8.6.2 結果

結果はかなり劇的なものとなった．NTP の場合，ホストはかなり正確であった．8 つのホストのみが 10 秒を越えるエラーを起こしたが，これらのホストは NTP の古いバージョンを用いているためと思われた．少なくとも NTP ホストの 30%が 30 ミリ秒以内の時間を維持していた．

ICMP と TIME は正確さにおいてはるかに劣る．ICMP ホストの 1%が 1 日より長いエラーを起こし，TIME ホストの 1%が数時間以上のエラーを起こした．ICMP ホストの 30%が数分間に及ぶ誤差を保持し，TIME ホストの 30%が 1 分以内の誤差を保持した．一般に NTP は他のプロトコルに対し 3 乗のオーダー（1000 倍）で正確な時計を維持する．

さらに研究が NTP 同期サブネット内で進んだ．18 か月の期間にわたり，観測は国中に置かれた 6 つの 1 次タイムサーバに対して行われた．これらサーバ間のパスはきわめて長い——いくつか

の場合には 12 のネットワークを経由(hops)する——こともあった．非常に多くのサンプルに対するエラーの平均はほんの数ミリ秒で，最大エラーは 50 ミリ秒にすぎなかった．

より進んだ勉強のために

D. L. Mills, Internet Time Synchronization: the Network Time Protocol. Forthcoming in *The IEEE Transactions on Communications.*

(掲載予定．louie.udel.edu 上の pub/ntp/trans.ps.Z より FTP で入手可能)

———, Measured Performance of the Network Time Protocol in the Internet System, RFC 1128, October, 1989.

———, Measured Performance of the Network Time Protocol in the Internet System, *Computer Communication Review* 20, 1 (January 1990), pp. 65-75.

———, Network Time Protocol (Version 3), Internet Draft Standard, December, 1990.

(louie.udel.edu 上の pub/ntp/ntpv3.ps.Z より FTP で入手可能)

Chapter 9

名前

9.1 新名称の名前づけサービス：~~Yellow Pages~~ NIS

Sun ONC プロトコルファミリーの元来の名前づけサービスは，イエローページ（Yellow Pages）と呼ばれていた．うっかりして，Sun のマーケティングはこの名前を本当に使ってよいかを調べ忘れてしまった——特に名前づけサービスでは何といっても他人の名前を使ってはいけない．そうでなければ，新しいコンピュータの系列を“MicroVAX VI”と呼んでしまえば簡単だろう．Yellow Pages の場合は，British Telecom 社が単刀直入に異議を唱えてきたようである．そこで最近のマニュアルではすべて，Sun の名前づけサービスはネットワーク情報サービス（Network Information Service : NIS）と呼ばれている．もちろん，コマンドは依然として“yp”の接頭辞から始まることが多い．

NIS が開発されたそもそもの動機は，システムファイルが増殖してしまうという問題であった．例えば，/etc/passwd を例にとって考えてみよう．たくさんの NFS サーバとたくさんのワークステーションのある分散環境では，ネットワーク全体で重複のないユーザの名前空間をもつことが有意義である．各コンピュータは独自の/etc/passwd ファイルをもっているので，重複のない名前空間をもつには，すべてのファイルを同期させる何らかの方法が必要になる．もちろん，これを実現するための当初の方法は，システム管理者が記述した複雑なシェルスクリプトで“公式な”コピーをネットワークの他のすべてのホストにコピーするものであった．各サイトは独自の方式をとっていて，ほとんどのものは大変な労力を費やしてやっと動くものであった．

NIS の役割は，名前保持のサービスを配布する機構を提供することである．基本的なサービスでは，キーを受け取り値を返す．例えば，キーはユーザ名で値は暗号化されたパスワードであろう．また，キーがホスト名で値がそれに対応する IP ホストアドレスであるかもしれない．

NIS が提供する各変換はマップとして知られている．いくつかのマップの場合は，実際には 2 つのサービスが存在する．ホストは，ホスト名をホストアドレスに変換したいこともある．例えば“user@host”という宛先のメールメッセージが与えられたら，クライアントはそのホストの

SMTP ポートのインターネットアドレスを見い出さなければならない．一方それとは逆に，クライアントはホストのアドレスをホスト名に変換したいこともある．ある IP 発信元アドレス上から SMTP コネクションを受けたときに，クライアントはメールのメッセージの“Received From”というヘッダ行に書き込むために，ホスト名を知ろうとするかもしれない．

NIS はマップをもって，それをネットワーク上のすべてのノードで使えるようにする．NIS は協調してマップの更新を行い，複数の NIS サーバが同一のマップの中から答を検索することができるような機構を提供する．この複製により，クライアントが検索の結果を得るときの利用可能率と速度がともに向上している．

本章では，まず NIS の説明から始める．次に，NIS の後継である NIS＋について見ていく．NIS＋は NIS のマップと互換性を保ちながら，NIS よりもずっと洗練されたサービスを提供する．次に，ドメイン名前システム（Domain Naming System：DNS）について述べる．

NIS は Sun や ONC のホストからなる 1 つのグループ内で使うためのものである．いろいろな面で，ワークグループや企業ネットワーク用に最適化されているが，NIS は非常に大きなネットワークでも使い得るものである．それに対して，DNS は Internet の分散化された，広域環境で使うためのものである．DNS は NIS や NIS＋ほどのサービスを提供はしないが，複数の管理ドメインが互いにサービスを提供し合うことを可能にしている．

9.2 NIS

NIS は基本的なシステム管理ファイルを管理するための，簡単な分散化されたネームサービスである．NIS はまた，公開鍵マップを用いることで，ネットワーク内で Secure RPC と Secure NFS が動作することを可能にしている．

NIS は，ある領域内のコンピュータの中で，名前に重複がないということをかなりゆるく保証する．また，NIS はクライアントがその名前に対応した値を検索できるようにしている．このゆるい唯一性の保証は重要なものである．NIS は分散データベースサーバではなく，一般化された検索サービスは提供しないし，複数の値を直ちに収束させることを保証しない．

ドメインとは単にコンピュータの集合である．ネットワークはいくつかのドメインをもち得る．例えば，Sun 広域ネットワークは，いくつかの異なるドメインから構成されている．ネットワーク全体を 1 つのドメインに入れてしまうこともできるだろうが，分割した方が名前空間の管理が簡単になる．NIS は他のドメインの検索も行うことができるが，デフォルトでは操作は現行のドメインに対して行われる．

マップとは NIS サーバが保持するデータである．ほとんどのマップはテキストファイルに基づいていて，特別な make ファイルを用いて NIS フォーマットに変換される．各々のマップにはマスタサーバがあって，これがすべての変更の責任を担っている．

マップには2次サーバをもつことができる．2次サーバはマップのコピーをもっているが，最新情報であるとは限らない．時間が経つと，マップのマスタコピーが2次サーバに伝搬あるいは“プッシュ”されることをNISは保証している．このゆるい整合性は，データが急速に変化しないような環境でNISを理想的なものとしている．“急速”というのはもちろん相対的な言葉であるが，ここではミリ秒や秒単位ではなく，分単位あるいは時間単位で変化するようなファイルのことを指している．

すべてのONSクライアントは，サーバにNIS要求を送ることによってこれらのマップを参照する能力をもっている．サーバを見い出すために，ypbindという特別なプロセスが用いられる．ypbindはサーバを探す責任を担っている．いったんバインドがなされると，マップを参照するためにいろいろなシステムコールやユーザのユーティリティが用いられる．

9.2.1 NISの構成要素

NISには次のような5つの基本要素がある．

・ドメイン（domain）

・マップ（map）

・デーモン（daemon）

デーモン	
ypserv	サーバプロセス
ypbind	バインディングプロセス
ypxfrd	マップ転送デーモン
rpc.ypupdated	マップのエントリ変更のためのサーバ
rpc.yppasswdd	NISパスワードファイル変更のためのサーバ
in.named	DNS環境で用いる付加的なデーモン
ユーティリティ	
ypcat	マップのデータをリストする
ypwhich	NISサーバとサービスされるマップの名前を列挙する
ypmatch	キーとマップ中の値とを照合する
ypinit	データベースを作成してインストールする
yppoll	サーバからマップの順番番号を得る
補助ユーティリティ	
yppush	マスタサーバからスレーブサーバへ伝搬する
ypset	特定のサーバにバインディングを設定する
ypxfr	マスタサーバからスレーブサーバへデータを転送する
makedbm	NISマップのためのdbmファイルを生成する

9-1 NISの構成要素

・ユーティリティ（utility）

・保守管理用コマンド（maintenance command）

ドメインは共有されるマップの集合にすぎない．各ドメインはドメイン名をもっていて，各コンピュータはただ1つのドメインに属している．

デーモンはクライアントやサーバの上で動作するプロセスであり，ユーティリティはクライアントで動作するプログラムである．保守管理コマンドは，ネットワーク管理者がマップを作成したり変更を伝搬させるのに用いる(図9-1参照)．ユーザコマンドだけでなく，プログラマはライブラリルーチンを用いてNISの機能を利用することができる．

9.2.2　マップ

マップはサーバにおいて

　/var/yp/domainname/...

というディレクトリに保持される．マスタサーバの/var/ypの中を見ると，マップを作るためにmakedbmを呼び出すmakefileがある．

makedbmは実際にマップを作るプログラムである．これは入力としてキーと値の組を受ける．キーは1つの単語で，値はその行の中で後に続く任意のデータである．この結果，以下の2つのファイルができる．

　/var/yp/domainname/mapname.dir

　/var/yp/domainname/mapname.pag

Sunネットワークで用いられるマップには，非常に多岐にわたるものがある．図9-2は，デフォルトのマップのいくつかを示している．これに加えて，プログラマやネットワーク管理者は他のアプリケーションのために新しいマップを作ることができる．

多くのマップが/etcディレクトリにあるファイルを置き換えたり補足したりしていることに注意してほしい．典型的な例としては，ホストファイルやパスワードファイルである．いくつかのユーティリティがNISのサービスの恩恵をこうむるためには，いくつかのUnixユーティリティを書き換える必要があった．例えば，passwdコマンドはユーザが自分のパスワードを変更することを許している．このコマンドは/etc/passwdファイルを編集することでそれを行っている．passwdコマンドはsetuidコマンドとして書かれている．これはファイルの1行を変更するためだけにスーパーユーザの特権をユーザに与えるものである．

/etc/passwdファイルがパスワードマップに置き換えられたときは，パスワードコマンドを書き換えて，それがファイルとNISマップの両方からパスワードを探すようにしなければならない．同様に，ログインプロセスも自ホスト内のファイルを調べるだけでなく，マップに対してNISコールを発行することを知るようにしなければならない．

ほとんどのマップはマスタサーバ上のテキストファイルの中に管理されている．マップの更新

マップ名	ソースファイル	記述
bootparams	bootparams	ルートやスワップのような，クライアントがブートするときに必要なファイルのパス名．/etc/bootparamsを置き換える．
ethers.byaddr	ethers	マシン名とアドレス．/etc/ethersを置き換える．
ethers.byname	ethers	上と同じだがインデックスがマシン名
group.bygid	group	グループセキュリティに関する情報で，グループIDがキー．
group.byname	group	名前がインデックス．
hosts.byaddr	hosts	ホスト名とIPアドレスで，IPアドレスがキー．/etc/hostsはブート時に参照される．
hosts.byname	hosts	上と同様でホスト名がキー．
mail.aliases	aliases	別名とメールアドレスで，別名がキー．自ホストの/etc/aliasesがまず調べられる．
mail.byaddr	aliases	上と同様でメールアドレスがキー．
netgroup.byhost	netgroup	グループ名とユーザ名とホスト名（ホスト名がキー）．/etc/netgroupを置き換える．
netgroup.byuser	netgroup	上と同様でユーザ名がキー．
netgroup	netgroup	グループがキー．
netid.byname	passwd, hosts,group	Unix流の認証に用いる．マシン名とメールアドレス(ドメイン名を含む)．netidファイルが先に調べられる．
netmasts.byaddr	netmasks	IPサブネットに用いるネットワークマスク．/etc/netmasksは調べられない．
networks.byaddr	networks	名前とIPアドレス．/etc/networksを置き換える．
networks.byname	networks	ネットワーク名がキーになる．
passwd.byname	passwd	ユーザ名がキー．/etc/passwdを補う．
passwd.byuid	passwd	ユーザIDがキー．
protocols.byname	protocols	知られているネットワークプロトコル．/etc/protocolsを置き換える．
protocols.bynumber	protocols	プロトコル番号がキー．
publickey.byname	publickey	secure RPCに用いられる．/etc/publickeyを置き換える．
rpc.bynumber	rpc	RPCに知られているプログラム番号と名前．/etc/rpcを置き換える．
services.byname	services	インターネットのサービス．/etc/servicesを置き換える．
ypservers	none	既知のNISサーバ．

9-2 NISのマップ

は，この入力テキストファイルを編集した後，make ユーティリティを実行すればよい．あるマップに関してどのサーバがマスタであるかを知るには，ユーザは次のように ypwhich コマンドを用いればよい．

```
mynode% ypwhich -m map_name
```

ここで，map_name はマップの名前かニックネームである．得られる結果は，ネットワーク上のホストの名前である．

多くの状況で，以下の 9 つの/etc ファイルが，NIS マップですべてあるいは一部置き換えられ得

る．

- passwd
- hosts
- ethers
- group
- networks
- protocols
- services
- aliases
- netmasks

passwd ファイルは NIS マップで補足されるべきで，必ずしも完全に置き換える必要のないファイルの一例である．プログラムはまず最初に自ホストの/etc/passwd ファイルを見てユーザを調べ，その次に NIS マップを調べる．各クライアントは，ルートと主要なユーザのエントリの入った局所的なパスワードファイルをもつべきである．“局所的”というのは相対的な言葉であり，ディスクレスノードではこのファイルは実際には NFS サーバにあるかもしれない．ファイルの最後に，NIS パスワードデータベースを探索させるために次のようなエスケープシーケンスが使われる．

```
root:9wxntql2tHT.k:0:1:Operator:/:/bin/csh
myname:7ljDXZD/Hug2s:624:20:myname:/home/myname:/bin/csh
+:
```

hosts.equiv も NIS マップで置き換えられるかもしれないファイルである．通常は，hosts.equiv はプラス文字だけが書かれた 1 行だけからなる．このエスケープシーケンスは，パスワードデータベースの中の人は誰でも，既知のどのホストからでもこのマシンにアクセスできることを意味する．これはデフォルトである．

netgroups を用いれば，より安全になる．hosts.equiv ファイルに＋@や-@を書き込むと，アクセス許可をどう扱うか決めるのに NIS ネットグループデータベースを使うようになる．

/etc/hosts ファイルは，各システムで独立に管理しなければならないものである．マシンが NIS サービスを使っていても，ローカルホスト名は必要である．ディスクレスマシンでは，このファイルにはブートサーバの名前も入っていなければならない．ypbind が(ブートが完了してから)起動され実行状態になると，/etc/hosts はもう使われなくなり，NIS のホストデータベースにとって代わられる．

groups などその他のファイルは，厳密に NIS マップとして簡単に管理できる．ユーザプログラムに，マップを調べるように教えるためには，クライアントの上ですべてのファイルを次のように簡単にしてしまう．

```
hydramatic% ypwhich
commserv

hydramatic% ypwhich -m
hosts.byaddr parkway
sg.auto.home parkway
sg.auto.db parkway
sg.auto.design parkway
mail.aliases parkway
APPCs.byname comic
sparcdudes.master parkway
auto.master parkway
auto.nse moon
timezone.byname parkway
netgroup.byuser parkway
appcs.byname parkway
resources parkway
networks.byname parkway
products parkway
netgroup parkway
netgroup.byhost parkway
rjes.byname parkway
auto.dump parkway
protocols.byname parkway
ypservers parkway
3270s.byname parkway
ethers.byaddr parkway
passwd.byuid moon
nse.branch.byid moon
ethers.byname parkway
nse.global.byname moon
passwd.byname moon
netid.byname moon
rpc.bynumber parkway
services.byname parkway
bootparams parkway
mail.byaddr parkway
hosts.byname parkway
group.bygid parkway
publickey.byname moon

hydramatic% ypwhich -x
Use "passwd" for map "passwd.byname"
Use "group" for map "group.byname"
Use "networks" for map "networks.byaddr"
Use "hosts" for map "hosts.byaddr"
Use "protocols" for map "protocols.bynumber"
Use "services" for map "services.byname"
Use "aliases" for map "mail.aliases"
Use "ethers" for map "ethers.byname"
```

9-3 NISの状態を知るコマンド

+:

デフォルトのマップに加えて，多くのサイトはその他の広範な情報をNISで管理している．図9-3は，NISの現在の状態を知るのに用いられるypwhichコマンドの典型的な出力例である．

最初のコマンド，すなわちパラメータなしのypwhichは，ワークステーションhydramaticのデフォルトサーバはcommservという名前のサーバであることを示している．次のコマンドは，特定のマップについてどのサーバがマスタサーバとして働いているかを示している．様々なマップが管理されていることに注目してほしい(しかも，この図は実際使われているマップのごく一部を示しているにすぎない)．そして，最後のypwhichコマンドは，いくつかのマップについて使われるニックネームを示している．

9.2.3 NISサーバ

NISサーバはNISの作業の大部分を行うシステムである．これにはマスタとスレーブという2つのタイプのサーバがある．マスタサーバは更新を担当して，それをスレーブサーバに伝搬する．スレーブサーバはマップの読出し専用コピーをもつ．

一般的な規則として，NISサーバはNFSファイルサーバと同じマシンであることが多い．これはもちろん必要条件ではないのだが，この2つの機能を組み合わせることは有意義である．同様に，一般的な規則として，バックボーンへのゲートウェイがNISサーバ，特にマスタサーバを兼ねるのはあまりよくない選択である．ゲートウェイの負荷が重くなるときは，ネームサーバが忙しいときと一致することが多いであろう．このため，この2つの機能を1つのマシンで行うと，ピーク時の問題を悪化させることになる．

スレーブサーバはマスタのマップの完全なコピーをもっている．スレーブが必要とされるのは以下の2つの理由による．

・スレーブを使うと，問合せの負荷が複数のシステムに分散するのでパフォーマンスが向上する．

・スレーブを使うと，重要なシステムファイルの複製を保持しているので利用可能率が向上する．

NISはブロードキャストを用いるので，各サブネットワークにはスレーブサーバかマスタサーバが最低1つはなければならない．これは，TCP/IP環境においてはブロードキャストはゲートウェイを越えて転送されることがないからである．典型的には，各ローカルセグメントで2つもしくは3つのサーバが稼働状態になっていて利用できるようになっている．もしも1つのNISサーバがアクセス不可能になっても，クライアントは自動的に他の稼働状態のサーバに結合づけし直すであろう．

9.2.4 結合づけ

結合づけとは，1つのサーバを選ぶプロセスである．ypbind がこれを扱うコマンドである．結合づけが確立すると，クライアントはそのサーバ上の ypserv デーモンに呼出しを行う．サーバとクライアントの結合づけは，ネットワークの負荷に応じて変更することができる．ypwhich コマンドにより，その時点でどのサーバが処理を行っているかがわかる．

```
my_node% ypwhich
yp_server
```

この結合づけ処理が，各サブネット上にサーバが必要となる理由である．ypbind は RPC ブロードキャストを発行するが，これにはサブネット上のすべてのホストに届く IP ブロードキャストアドレスが格納されている．クライアントの要求に答えた最初のノードがそのクライアントのサーバになる．もし何らかの理由でサーバが利用不能状態になると，ブロードキャストが再び行われて，新しいサーバが選ばれる．

結合づけはサブネットワーク内に限られるが，だからといって NIS のドメインがサブネットワーク内に限られるわけではない．マップの受渡しの処理はブロードキャスト機構を用いないので，サーバどうしはゲートウェイ境界を越えて通信できる．

9.2.5 システムの初期設定

マスタサーバの起動は，単に初期化スクリプトを走らせてからサーバを起動するだけのことである．一般的には，これらのコマンドは rc.local に挿入されて，マスタサーバがブートされるといつも実行される．

```
# cd/var/yp
# /usr/etc/yp/ypinit -m
# ypserv
```

マスタサーバ上では，ypxfrd という高速転送デーモンも起動されるであろう．

スレーブサーバは，同じステップを走らせる必要があるが，その前にまずマスタサーバの名前を知る必要がある．

```
# cd/var/yp
# ypbind
# /usr/etc/yp/ypinit -s master_name
# ypserv
```

クライアントのシステムはいたって単純である．ユーザはどこかに ypbind コマンドを置きさえすればよい．これでクライアントは適切なサーバの名前を見つけることができる．

一般的には，NIS はユーザにとって目に見えないものである．1つの例外は passwd コマンドで

ある．ユーザが自分のパスワードを変えるときは，マップを変えようとしているのである．特別なyppasswdコマンドが新しいパスワードをマップの中に作ってくれる．これがうまくいくためには，マスタサーバでrpc.yppasswddデーモンが動いていなければならない．

マスタのマップを更新することと，スレーブのマップのコピーにその変更を伝えることは別のことである．一般的には，ユーザは家に帰る前にこれらの変更をするべきだろう．更新情報は翌日までには完全に伝搬されているだろうから．

マップの伝搬は，定期的にのみ行われる類のものである．スレーブがypinitを行うとき，マップのすべてが強制的に転送される．その後は，ypxfrコマンドを実行する必要がある．このコマンドは通常はサーバのcrontabファイルに記入される．こうすると，それは定期的に，通常は毎晩実行される．

各スレーブサーバのcrontabにはypxfrエントリの列がある．ypxfrは起動されるたびにマスタサーバと連絡をとって，マスタがより新しいマップのコピーをもっているときのみ転送が行われる．

```
/usr/etc/yp/ypxfr -h master mapname
```

ypxfrコマンドはyppushと呼ばれる別のコマンドと対になって働く．yppushはypserversというマップを調べ，その中の各スレーブサーバと連絡をとり，“transfer map”要求を送る．要求がスレーブに受理されると，スレーブはypxfrプログラムを用いて実際の転送を行う．

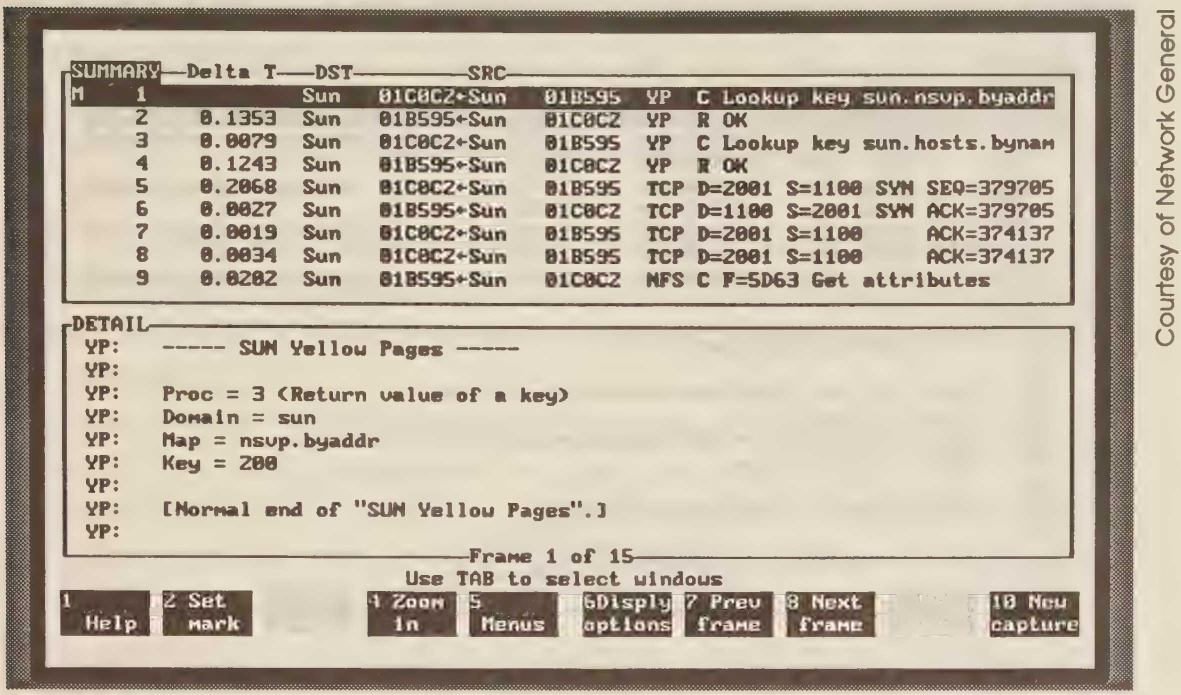
新しいマップの要求をクライアントに任せるということは，スレーブがダウンしているときでもシステムは働き続けることを意味する．マスタがダウンしているときには，スレーブは新しい変更を受け取ることができないが，古いままのデータをサービスし続けることができる．

ypxfrコマンドは-sという特別なオプションをもっている．このオプションにより，マップを別のドメインから受け取ることができる．これはSunが“マスタ・マスタ”データベースを扱ってきた方法である．あるドメインのマスタは“マスタ・マスタ”にアクセスして，中央で管理されているファイルを得る．このようにして，各ドメインは別々のマスタサーバをもちながら，マスタサーバどうしが協調して働くことができる．

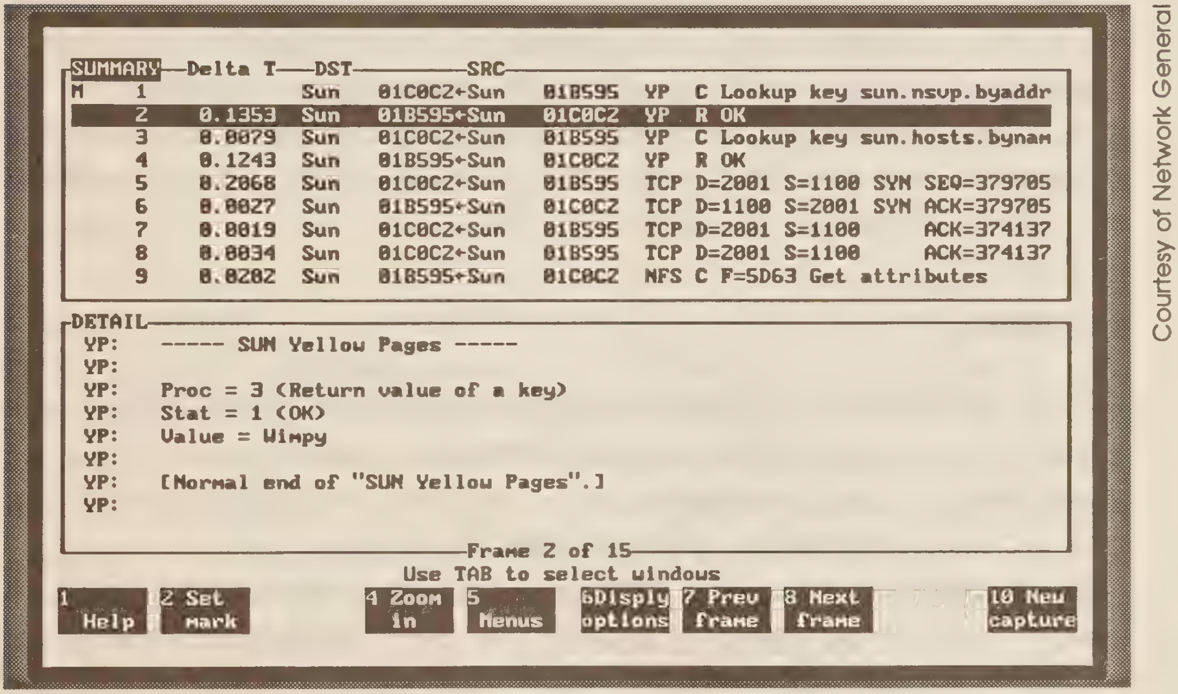
9.3　典型的なNISの動作

図9-4から図9-6までは，NISサービスの典型的な動作を示している．図9-4において，最初の操作はSunドメインにおいてアドレスをキーにした検索である．図9-5はその結果であるWimpyという名前を示している．

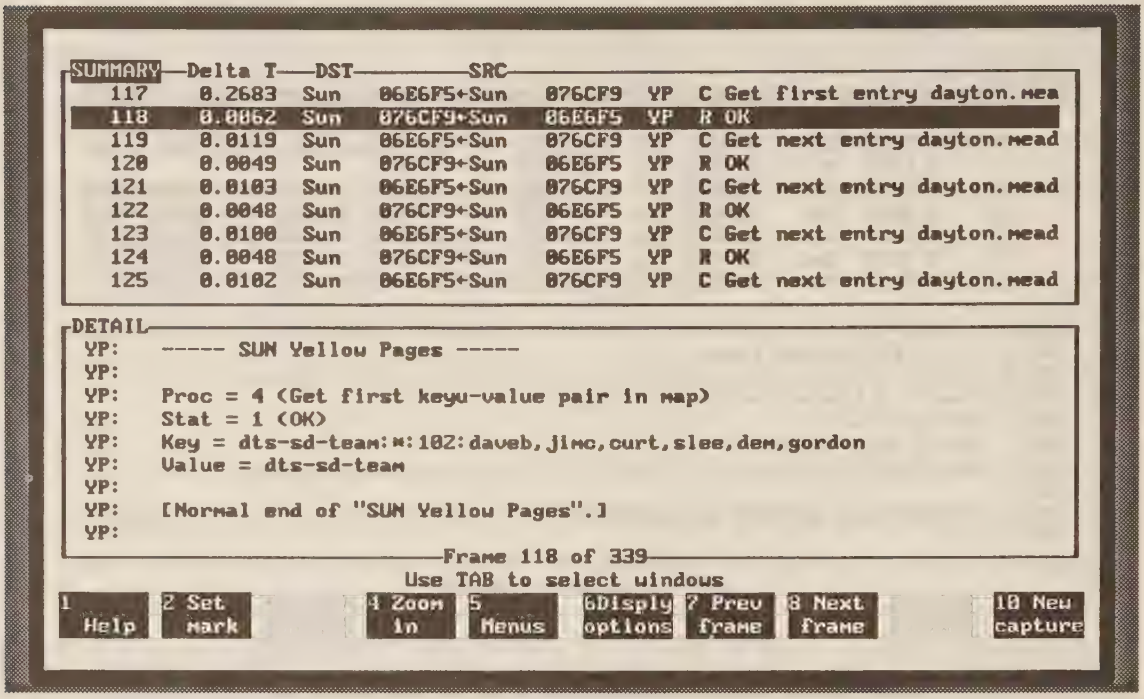
図9-5では，この続きとして，ホストWimpyのインターネットアドレスを調べるためにhosts.bynameマップの検索を行っているのが見られる．図9-5によれば，このアドレスが見つかった後，クライアントノードはWinpyにTCPコネクションを設定している(次の2つのTCPパケッ



9-4　NISの問合せ



9-5　問合せの結果



Courtesy of Network General

9-6　強力なget next操作

トで同期ビットがセットされていることに注目してほしい)．パケット番号9を見ると，これらすべてのトラフィックの理由がわかる．ファイルの属性を得るためのNFS動作が行われている．

最後の図9-6では，検索を繰り返すマップ操作の例，この場合はDaytonと呼ばれるドメインでのグループの検索の例を示している．強力なget next操作により，1つのマップに対して繰り返して問合せを行うことができる．後ほど，NISの後継であるNIS+が，似たような1行ごとの検索をずっと柔軟性のある方法で行っていることを見るであろう．

9.4 NIS+

NISは，それに割り当てられた比較的小さなネットワークで使われる，共通システムファイルを管理するという，限定された役割にはきわめて効果的である．NISはもっと大きなネットワーク（最も特筆すべきなのは，Sun内部のネットワークの10 000ノード）にまで何とか対応してきたが，いくつかのより現代的なサービスのための機構に欠けている．

NISの後継者はNIS+である．NIS+は軽量な(lightweight)ネームサービスである．これは，本格的なデータベースではないということを意味している．つまり，これはX.500と相補うサービスである．X.500は属性に基づく成熟したネーミングシステムである．NIS+は，より限定された一部分の機能をX.500よりずっと高速に提供するためのものである．NIS+の変更点を理解する

ために，まずはNISの4つの制限から説明を始めよう．

・フラットな名前空間

・オブジェクトが文字列に限られている

・最低限のアクセス制御と認証(authentication)しか行わない

・更新の能力がない

フラットな名前空間というのは，マップがドメイン全体に公表されることを意味している．また，サーバが他のドメインの情報を知ることができないことも意味する．これには，名前空間の部分集合をもつサブドメインのための用意がない．NIS+は名前空間に階層を付け加えることでこの問題を解決している．

NISでは，オブジェクトの値は単純な文字列である．NISのもともとの目標には十分であるが，クライアントがマップに任意の種類のデータを入れておきたい場合もある．

NISはアクセス制御をもたない．NISの基本的なサービスはデータへの読出し専用アクセスで，セキュリティのための用意がない．すなわち誰でも何でも読める．非公開マップやマップのアクセス制御リストといった概念はない．例えば，パスワードマップを例にとると，ユーザはすべての人の（暗号化された）パスワードを見ることができる．これは必ずしも大きなセキュリティホールではないが，システムを確実に侵入者から攻撃されやすくしている（この特定問題の解決策については，Sunオペレーティングシステムの C-2 セキュリティ強化で提供されている）．

最後に，NISは管理が難しいという問題をもっている．サービスは検索サービスのみであり，データの変更のための用意はない．このため，変更を行うためには，マスタサーバにログインして，テキストファイルを編集し，そのあとでmakeユーティリティを走らせなければならない．

まずはNIS+の簡単な概要から始めて，そのあと各項目について詳細に述べる．NIS+は次の2つの基本的なサービスを用意している．

・分散ネームサービス

・データベース管理システム

名前空間の提供者として，NIS+はユーザが名前を付け加えたり，取り除いたり，検索したりすることを許している．検索操作は，名前をオブジェクトの特定のインスタンスに変換する．

NIS+が操作するデータベースは階層的な名前空間の中に名前づけされている．データベースの提供者として，NIS+はテーブル中のレコードの操作のためのadd, remove, searchといった一連のプリミティブを提供している．ここで，テーブルは名前空間で名前づけされたオブジェクトであるが，レコードには直接は名前がつけられていないことに注意されたい．

NIS+のサービスは一連のネームサーバにより提供される．各サーバは名前空間の一部を処理する．ルートサーバを例外として，サーバはその下にある名前空間を扱う．

各ネームサーバのデータベースの内容は，1つもしくはそれ以上のサービスポイントに複製することができる．サービスポイントの中の1つはプライマリサービスポイントであり，名前空間に

変更を加えたいプリンシパルの連絡先である．

NIS＋のクライアントはプリンシパルと呼ばれる．各々のプリンシパルは公開鍵と秘密鍵の組をもっていて，認証のために用いられる．あるプリンシパルに関する情報をNIS＋からアクセスするには，プリンシパルの名前と，その公開鍵を必要とする．NIS＋へのエントリポイントはそのプリンシパルに定義されたアクセス権限でさらに制限されるかもしれない．

問合せの対象がプリンシパルであるのと同様に，問合せの発信元もプリンシパルである．発信元，つまりあるマシン上のユーザ名は，名前および先ほどと同一なキーの組で識別される．ホストはこの情報を，ユーザに関する情報を検索するのに使うことができる．例えば，ログインのためにユーザ名に対応する暗号化されたパスワードを検索することができる．

9.4.1 名前

NIS＋の名前はラベルの列で構成されている．各ラベルはディレクトリツリーの一部に対応している(図9-7参照)．階層の頂上はルートと呼ばれる．ルートはディレクトリと呼ばれるタイプのオブジェクトである．ディレクトリは，その内容として一連の名前をもっており，それぞれがオブジェクトになっている．

ルートの下にはまたディレクトリがある．最終的には，木の底(葉)に，別の種類のオブジェクトがある．これはファイルシステムに類似している．一連のディレクトリがあって名前が終わりにある．

NIS＋では，木の葉には様々な異なる種類のオブジェクトがある．そのうちの1種類は表であり，これはNISのマップと等価なものであるが，おおいに拡張されている．これ以外の種類のオブジェクトもある．例えば，シンボリックリンクはネームツリーの別の部分を指すポインタで，別名を提供する手段となっている．

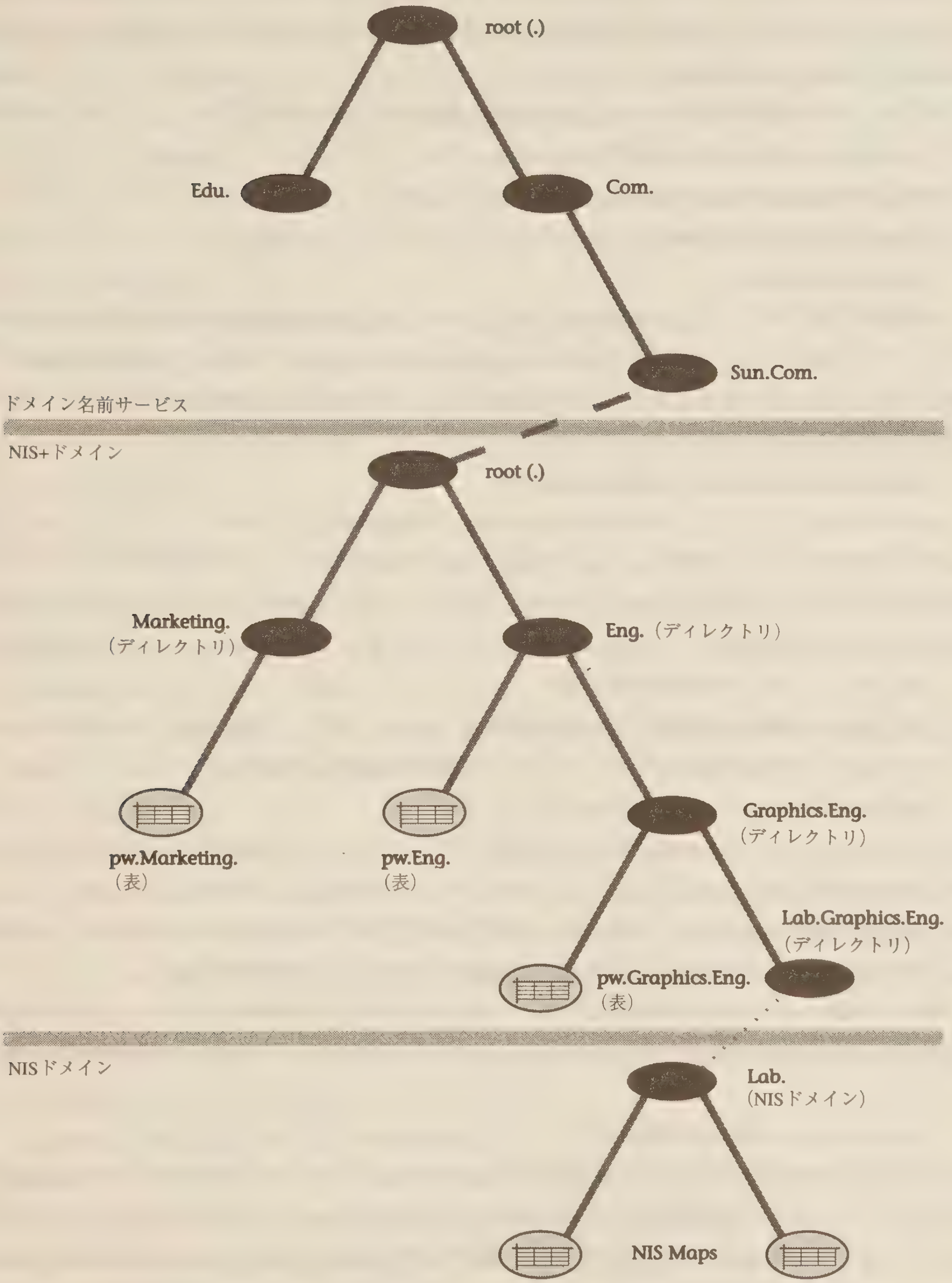
NISでは，すべてのオブジェクトは共通のドメインにある．デフォルトではどの検索も現在のドメインに関するものである．もっとも他のドメインを検索することも，名前を知っていれば可能である．これは1つのルートと1つのマップというフラットな名前空間の例である．

NIS＋では名前空間を組織化できる．ルートがEngineeringであるとしよう．ルートの下にサブドメインがある．

```
Graphics.Engineering
FDDI.Engineering
Database.Engineering
```

Graphicsはディレクトリであるから，その中に一連のオブジェクトをもつ．これらのオブジェクトは，passwd, hostname, groupsやその他の管理ファイルのような従来のNISのマップかもしれない．

```
Passwd.Graphics.Engineering
```



9-7 NIS+と他のネームサービス

がその例である．サブドメインの数は非常に大きくすることができる．Graphics が最終的なサブディレクトリではなくて，個人に対応した低レベルの名前があってそれぞれが固有のデータベースをもっているかもしれない．

オブジェクトは階層のどのレベルにでも存在し得るので，各々のサブディレクトリがパスワードファイルをもっているかも知れない．

Passwd.Malamud.Graphics.Engineering

Passwd.Graphics.Engineering

Passwd.Engineering

名前空間に関するもう 1 つの特徴を述べよう．ルートが実際には他のネーミングサービスへのポインタかも知れない．これまで Engineering はルートドメインとして使ってきた．Marketing もまたドメインであるべきだと思われるかもしれない．これらはともに組織全体の名前システムの一部となり得る．

Passwd.Malamud.Graphics.Engineering.Sun

Sun の下のオブジェクト，すなわち Engineering と Marketing は，別のネーミングシステムである DNS で管理される．実際には，Sun も Sun.Com の短縮形であり，名前の階層のレベルを 1 つ付け加えているものである．Internet の中央ネームサーバがトップレベルのディレクトリである Com を扱う．別の DNS サーバが Sun のサブドメインを扱う．Engineering のレベルまで降りると，NIS＋ネームサーバが名前空間を扱う．

このような長い階層の概念は，長い名前を助長しそうに見える．現実には，ほとんどのシステムがデフォルトの概念を用いている．ローカルな検索の場合は，例えば，

Malamud.Graphics.Engineering.Sun.Com

がデフォルトになる．“passwd” はローカルなパスワードファイルを検索し，“passwd.FDDI” を問い合わせれば，FDDI サブディレクトリのファイルを要求することになる．

この階層の強力なところは，非常に広い環境の名前をアクセスできるという点である．自分のところのネームサーバが扱えない場合は，木の上の方のサーバに要求がなされる．

NIS＋と他のシステムとの領域をどのように切り分けるかはその機関に任されている．今の例では，Engineering が NIS＋の扱うもののルートである．例として，次のような問合せ対象があるとしよう．

Passwd.Hyperbole.Marketing.Sun

自ホスト内の NIS＋サーバはこの問合せを Engineering サーバにまでさかのぼって送る．そこで Sun ディレクトリは範囲外であることがわかる．Engineering サーバはクライアントプログラムにメッセージを送り返して，Sun ドメインは DNS で扱われることを示す．DNS は Marketing.Sun サーバの場所を教えてくれる．その後，クライアントはまた NIS＋に切り替えて，Hyperbole サブドメインでの検索を試みる．

NIS+とDNS(およびX.500やその他のシステム)を切り替えるためには賢明なクライアントが必要である．もう1つの選択肢として，NIS+をできる限り使うようにする方法もある．例えば，DNSとNIS+との間の線をSunの上で引いてみよう．DEC.COMに関する要求はすべてDNSを参照する．Sun.Comの名前空間の中での処理はNIS+が扱う．

このように，名前はドットで区切られたラベルの列である．ラベルはピリオド，ダブルクォート，コロン，パーセント，角括弧のような特殊な文字を通常含まない文字列である．NIS+の名前からコロンやパーセントが除かれているので，DigitalのDNAネーミングサービスと共存できる．詳しくは後述する．

もし特殊文字が名前の中に必要な場合は，ダブルクォートを用いる．

"Any.Bad%Awful : Name".Hyperbole.Marketing

このクォートはラベルの中身を名前サービスから隠す働きをする．このような名前は，異なるネーミングサービスの文法を混ぜて使うことから生じる多くの特殊事情や曖昧さがあるため，特によくないものである．

9.4.2 オブジェクト

最終的に名前空間の階層の末端にたどり着くと，そこにあるのは基本的なオブジェクト，すなわちNIS+のデータベースオブジェクトである．NISのマップは，1つのキーが単一の値をもつような特殊な構成のデータベースである．

NIS+ではマップは表になっている．表は決まった数の欄と可変の数の行をもった構造体である．欄のうちのいくつかは検索が可能で名前をもっている．それ以外は単にデータを保持しているだけである．

NISのマップは実際には欄が2つの表で，一方の欄は検索が可能なものである．NISの世界では，両方の欄を検索可能にするには2つのマップが作られる．hostnames.byaddr✢がその例であり，1つのマップはアドレスで検索するために作られ，もう1つのマップは名前で検索するために作成される．この2つ組のNISのマップは，両方の欄が検索可能なNIS+の1つの表に置き換えることができる．

表の各行を参照するには，インデックス名(indexed name)の考えが用いられる．インデックス名はその欄の名前と検索すべき値である．

[username = Malamud],Passwd.FDDI.Graphics

検索基準によっては，1つ以上の行をもたらし得る．もちろん，ここにあげた検索は，ただ1つのユーザ名(とその暗号化されたパスワード)をおそらくもたらすであろう．

データベースの検索は基本的なNISの操作と等価である．それに加えて，NIS+ではオブジェク

✢(訳注) hosts.byaddrとhosts.bynameが正しい．

トを生成したり削除したり変更したりすることによって，ユーザがデータベースを更新できる．

オブジェクトはディレクトリ，表およびポインタである．NIS＋の中で，それぞれのオブジェクトは一連の共通の属性と，各タイプのオブジェクトに固有な可変部分とをもったデータ構造で表現される．

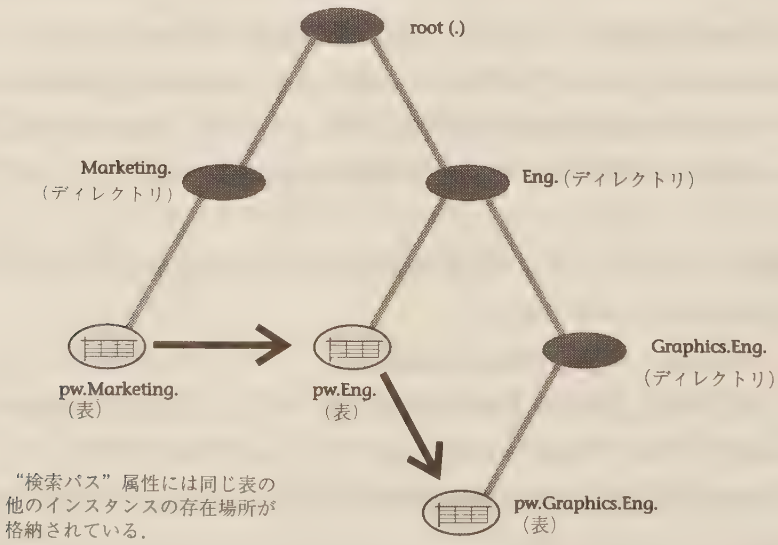
共通の属性とは，すべてのオブジェクトについてわれわれが知りたいような事項である．これらの共通の属性の例としては，そのオブジェクトがどの所有者やグループに属しているか，オブジェクトの生成時刻，最終変更時刻，アクセス権限，オブジェクトの種類といったものがある．

可変部分はオブジェクトの種類による．表の場合は，可変部分には，欄の名前（検索可能な場合）およびエントリのオブジェクトの種類と構造から構成される．

表の内部には一連のセルがあり，行と列の交差を表現している．各列はNIS＋のオブジェクトであり得るので，表の中にディレクトリを入れることも，表の中に表を入れることさえもできる．与えられた1つの列の中では，すべてのオブジェクトは同じ種類である．

表は末端オブジェクトの基本的な型であるが，表の行も列もNIS＋の名前をもたず，表自身だけがNIS＋の名前をもっている．行はインデックス名を使ってアクセスできる．1つのエントリ(例えばセル)は実際にはオブジェクトであり，オブジェクトと同様なデータ構造をもっている．すなわち，その固定部分に所有者や種類，構造情報といった情報をもっている．

表のすべての列が名前をもっているわけではない．いくつかの列は検索ができないものであり，それらに対しては名前は省略可能である．これが，NIS＋が汎用データベースと違う点の例である．汎用データベースでは表のすべての欄が常に名前をもっている．このように一般性を取り除くことによって，NIS＋は本格的なデータベースシステムよりも速くなっている．それは，真に任



9-8 NIS+の表と検索パス

意な検索をサポートするオーバーヘッドをもたないからである.

一般に,表のエントリは1つのデータかリンクオブジェクトのどちらかでできている.リンクオブジェクトは単にNIS+の名前であり，名前空間の別の部分へのポインタとなっている．また，検索パスを用いて一連の表をつなぎ合わせることも可能である(図9-8参照). 1つの表が検索されたあとで，別の表が指される．これはパスワードデータベースのように，データベースの細分化が必要なときにとても便利である．ユーザはパスワードに対して，共通な，ネットワーク全体にわたるビューをもちたいのに対し，ルートなどのユーザ名は各マシンで独自に管理する必要があるので，細分化されたデータベースが必要である.

オブジェクトの種類には，他にもう1つグループがある．グループはNIS+のプリンシパル(NIS+を使うコンピュータで名前をもっているもの)とそれに対応するグループの集合である．グループは再帰的にプリンシパルの集合に変換することができる(同じグループに再帰的に変換されないことが期待されている).

9.4.3 NIS+のサービス

NISでは，キーに対応した単一の値を検索するという，ただ1種類のサービスだけがあった. NIS+にははるかに多くの操作がある.

・名前空間に関する操作
・表に関する操作
・グループに関する操作
・複製に関する操作
・ナビゲーション
・オブジェクトの検証

名前空間に関する操作は名前空間自体を対象とする．最も一般に使う操作は検索操作である．ユーザはNIS+に名前を与え，NIS+はオブジェクトのコピーを返す．ディレクトリの場合は，NIS+の名前の組が返ってくる．それらの名前を検索すると，さらにディレクトリ，表，リンク，グループといったオブジェクトが返ってくる.

表に関する操作としては，ユーザが1つ以上の属性をキーとして指定して表を検索することができるコマンドであるリスト操作(list operation)が特徴的である．他のコマンドとしては，表を追加したり，削除したり，変更したりできるものがある.

NIS+のクライアントは，リスト操作によって複数の行が結果として返ることを認めている. NISでは,一度には1つの行しか返ってこない.互換性を保つために, first_ibaseおよびnext_ibaseという操作が用意されている.

グループに関する操作には，グループのすべてのメンバーを返す列挙操作がある．別の有用な操作としては，is_memberというプリンシパルがグループのメンバーであるか否かを確かめるも

のがある.

複製に関する操作により，サーバは情報をあちこちに移動することができる．receive 操作は 1 次サービスポイントと 2 次サービスポイント(NIS の用語ではマスタとスレーブ)との間で更新情報を転送する．ping 操作は，他のシステムに更新を知らせる．dump 操作は，ディレクトリ全体を複製する．

NIS と NIS＋との大きな違いの 1 つは, NIS＋がマップ全体を転送するのではなく, 更新情報をインクリメンタルに送ることができることである．これにより，更新の伝搬が，特にマップが大きくなったときに，NIS よりもずっと迅速になっている．

ナビゲーション操作は大きな名前空間のときに便利である．find_directory 操作は，名前を受け取ってネームサーバの位置を返す．これは，デフォルト以外のドメインのオブジェクトに対して通常行われる最初の操作となる．

最後の操作は，ノードがオブジェクトのキャッシュをもっているときに一貫性を保つために用いられるものである. is_newer 操作は 1 つのオブジェクトの 2 つのコピーを比べて, どちらが現在のバージョンであるかを判断する．

検索コマンドは実際には 4 種類ある. find_directory コマンドはサーバを見い出す. lookup コマンドはサーバが見つかってから，オブジェクトを返すのに使われる. list コマンドは表の中で用いられ，列挙操作はグループ内で用いられる．

9.5 NIS＋のセキュリティ

NIS＋へのアクセスは Unix ファイルシステムのモデルと同じ一般的な概念に基づいている. 所有者，グループ，全員に対して許可あるいは拒否する決まった数のアクセス権がある．

すべてのデータ（NIS＋のオブジェクトまたは表のエントリ）にはアクセス権が対応している. NIS＋のデータに関しては，生成，消去，変更，読出しの 4 種類のアクセスがある．所有者，グループ，全員に対して，これらの各種のアクセスが適用される．

読出しアクセスは，オブジェクトを見い出してその値を読むことができる．生成は，現在のオブジェクトが管理している名前空間に，新しいオブジェクトを追加できる．消去は，現在のオブジェクトが管理している名前空間の一部を消去することができる．変更はオブジェクト自身ではなく，その内部に適用される(オブジェクトの値を変えることはできるが消去することはできない).

表はエントリに対するアクセス権を列単位で用意している. これは, あるユーザが 1 つの列を読むことはできるが他は読むことができないかもしれないということを意味する．ユーザがある列に関して読出しアクセス権をもっていなければ，NIS＋はその列をマスクする(何の値も返さない).

アクセス権は，あるユーザがある1つのオブジェクトに対して何ができるかを示している．しかし，アクセス権を認めるより前に，ユーザを認識できなければならない．NIS＋では，認証の概念はプリンシパルに対して適用されている．これはホストやプログラムが含まれるので，ユーザという概念よりも一般的な概念である．

NIS＋における認証は，公開鍵暗号化システム(public key cryptosystem)を用いたSecure RPCに基づいている．セキュリティに関しては10章でより詳しく議論するが，基本的な考え方は，各プリンシパルに一組の信任状を与えることである．証明書は次の3つの部分をもつ．

・プリンシパルの名前

・公開鍵

・非公開鍵

非公開鍵はプリンシパルのパスワードを用いて暗号化されている．ユーザの公開鍵を用いれば，そのユーザだけが読めるようにデータを暗号化することができる．公開鍵で暗号化されたものを復号化できるのは，非公開鍵だけである．

逆もまた真である．プリンシパルが秘密の非公開鍵で何かを暗号化すると，そのメッセージを公開鍵で正しく復号化することで，それが本当にそのプリンシパル本人であることを検証することができる．

ユーザの信任状を取り出すには，名前の列を検索する列として使って，次のようなインデックス名を参照する．

[name =プリンシパルの名前],cred.プリンシパルのディレクトリ

このようにNIS＋では，信任状を取り出すことによって2つのプリンシパルが互いを認証し合うことができる．2つのサーバが互いに働いてよいことが検証されると，将来のトランザクションで使うために対称鍵(公開/非公開鍵よりもずっと高速である)を交換する．

このシステムはディスクレスワークステーションでも働き，リモートブートのセキュリティを大幅に増すことができる．ディスクレスマシンの非公開鍵はEEPROMに暗号化された形で格納される．サーバは，ワークステーションの非公開鍵で暗号化された情報とサーバのもつ公開鍵を比べることで，このワークステーションが正当なクライアントであることを検証することができる．

サーバが正当であることを検証するために，各マシンは，信頼されるネームサーバのネットワークアドレスとそのサーバの公開鍵をもったNIS＋のコールドスタートファイル(coldstart file)をもっている．これを用いて，サーバからのメッセージが本当にそのサーバからのものであり，他のシステムがそのサーバに変装しているわけではないことを検証することができる．このシステムは，2つのシステムが相互に認証し合うのには大変よく働くが，他の形のプリンシパルを検証するための別の方法が必要である．そのプリンシパルとはユーザである．

残念ながら，ユーザはEEPROMをもっていない(スマートカード(smart card)ならもっている

かもしれないが)．スマートカードがもっと普及するまでは，ユーザは2つの段階を踏んで認証される．まず，ユーザはログインの機構を用いて近隣のシステムにログインする．次に，その近隣のマシンがNIS＋サーバに対して仲介の役割を果たす（この時点で，ローカルマシンはすでに信頼されている)．近隣のマシンはネームサーバからユーザの信任状を検索し，ユーザのパスワードを用いてそれを復号化する．信任状を手にすることで，ユーザはネームサーバに対して認証された．

この結果得られるのは，各ユーザに代わってマシンが保持している安全な信任状である．マシンをログアウトするとき，マシンがもっている信任状を消去するように命令できる．このような認証処理は，信任状の期限が切れるまでの間，プリンシパルどうしで行われる．

信任状の期限が切れると，また認証が繰り返されなければならない．検索に対してたくさんの“……を見よ”が返されたときのように名前空間をトローリングする場合も，認証の繰返しを引き起こす．これは，オブジェクトの値を実際に検索する時間に比べて遅くなり得るが，それほど頻繁には起こらないはずである．

9.5.1　名前空間の複製

どのオブジェクトも1次サービスポイントをもち，更新はそこで行われる．更新は不可分に起こり，逐次化されていることが保証されている．更新が完了した後に，1次サービスポイントはそのオブジェクトを2次サービスポイントに伝搬させることができる．

NISと同様に，1次サービスポイント(マスタ)は，更新が起こったことを2次サービスポイント(スレーブ)に知らせる．そのあと更新情報を要求するのは2次サービスポイントが決めることである．これにより2次サービスポイントはどのくらいの頻度で実際にデータをもらうかを決められることになる（例えば，10分ごと，3回更新が起こるたび，10秒後，その他)．

1次と2次の間の更新はTCPを通して行われる．トランザクションが完全なときのみ，2次は複製を更新する．トランザクションは要求からレスポンスまでの完全な1回のRPCである．

プライマリが失われると更新が行えなくなる．しかし，プライマリサービスポイントを割り当て直すことができる．これは，問題のサーバの親ディレクトリに対して変更アクセス権をもっているプリンシパルにより行われる．

NISでは，データの“真実”はソースとなるテキストファイルにあった．NISはその真実のあるバージョンであり，makedbmが実行されないと，そのマップはもう古いものである．NIS＋ではデータベースが真実である．テキストファイルはどれもデータベースのダンプにすぎない．

データベースの複製が複数あるので，真実の“最も真実な”バージョンがある．それが1次サービスポイントである．しかしながら，他のすべてのコピーも十分似ている．もし，1次サービスポイントが死んでしまっても，それほど重大なデータの損失にはならない．失われるのはマップ全体ではなく，伝搬されていない更新だけである．

9.5.2 以前との互換性

NIS+の設計上の基本的な仮定の1つは,改善された環境の中で,現存するNISのクライアントが変更なしに動作を続けることができるようにすることであった.これはつまりNISクライアントが,相手がNIS+サーバであることを知らずに対話することができるということである.

NIS+では,NISのドメインは局所的なディレクトリである.NISのマップは2つの列からなり,その一方が検索可能なNIS+の表である.もちろんNIS+の1つの利点は,hosts.byname とhosts.byaddr のような双対のマップを1つのマップに入れることができる点である.

管理用のツールを用いて,NISのマップをNIS+の表に変換できる.忘れてはならない重要な点は,NIS+がNISをエミュレートするときにはセキュリティがないということである.マップに読出し許可がなければ,NISユーザに対して「該当するマップがありません」ということを示すエラーメッセージが戻ってくるであろう.

以前との互換性を保つため,ほとんどのNISクライアントライブラリはNIS要求を出し続けるか,内部でNIS+の要求に変換する.

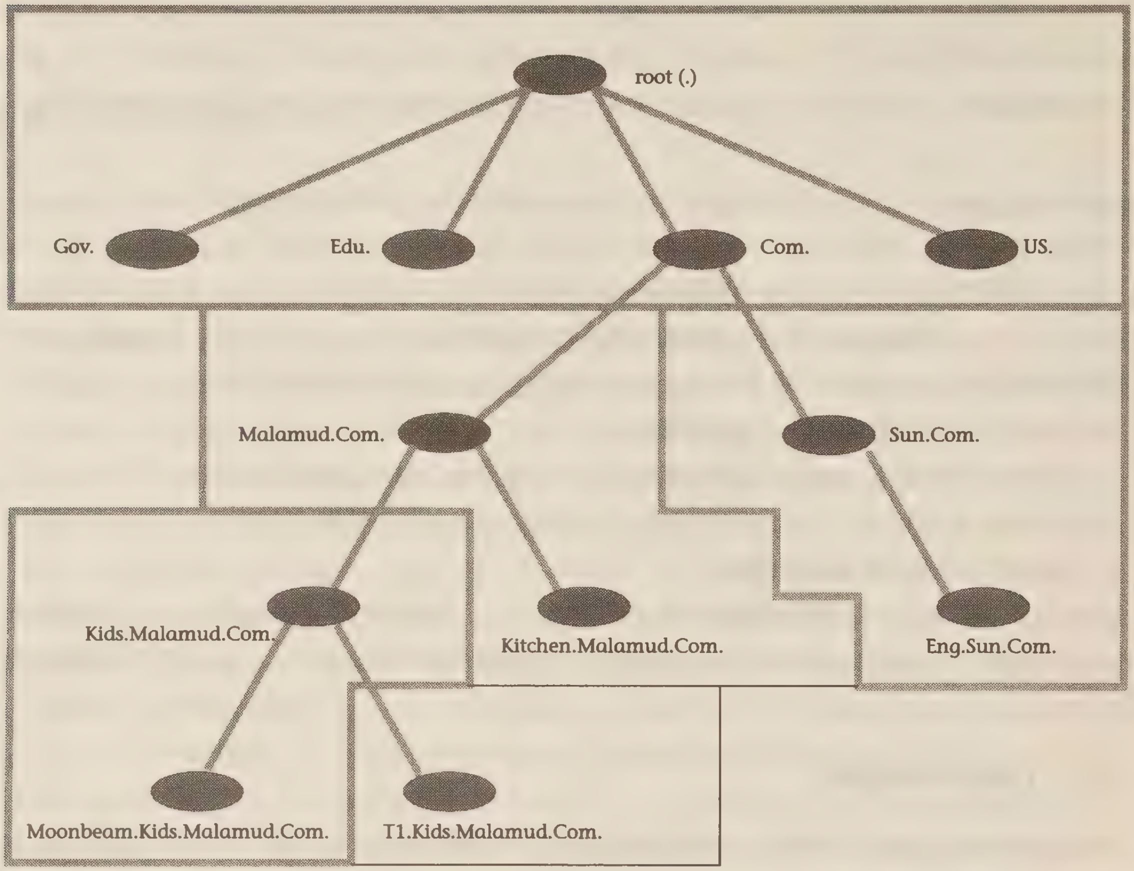
NIS+の木の末端がNISが管理するドメインであり得るように,木の頂上はしばしばドメイン名前システムのサービスである.例えば,NIS+のディレクトリ Foo.Sun.Com はDNSの親 Sun.Com をもつかもしれない.Bar.Sun.Com に対するNIS+の要求は次のようなメッセージを返す.

```
foreign ns == DNS, server == x
```

この時点で,なぜNISの代替としてDNSを使わずに,また新しいネーミングサービスを付け加えたのか,疑問に思う読者もいるであろう.1つには,NIS+はNISとの後方互換性があるので,現在のアプリケーションがうまく移行できるからである.より重要な点として,DNSはNISの代わりとしては不十分であるような問題点がある.まず,DNSにはセキュリティが存在しない.データに関するアクセス制限のない読出し専用ネームサービスなのである.NIS+は(少なくとも)個々の表にまでアクセス権を必要とする.さらにDNSはNISと同様に読出し専用のシステムである.DNSのデータの管理は,領域のマスタファイル全体の転送でなされるので,更新情報の伝搬が難しい.

DNSで扱えるデータはASCII(またはASCIIにエンコードされたバイナリ)のみに限られている.パケットサイズの制限もあるので,身内の環境で必要になりそうな比較的大きなオブジェクトは,DNSでは実際には格納することができない.もちろん,身内の環境では不都合なこの制限こそが,広域の環境で,広い互換性と効率のよい操作を保証するのに役立っている.

DNSが行ったことは,別々に管理された(しかし協調し合う)ネームサーバによって,階層的な名前空間が実現し得ることを示したことである.この概念はNIS+でも使われているが,局所的に管理された環境で必要な自由度と組み合わせられている."局所的に管理された"というのはもちろん相対的な言葉であり,DNSのようなシステムが扱う百万という単位に対して,数万から



9-9　DNSゾーン

CNRI.Reston.VA.US

あるグループをどのドメインに登録するべきかを定めるきちんとしたルールはない．例えば，著者はMalamud.Comに登録されている．しかし，個人は最上位ドメインに登録するべきでなく，適当なドメイン名は以下のようになるべきだと主張する人もいるかも知れない．

Malamud.SanFrancisco.CA.US

基本的なドメインに加えて，その他いくつかの最上位ドメインがある．Orgは，他のカテゴリーに分類できない組織に用いられる．調査会社であるMitreは，Mitre.Orgとして登録されている．ARPAドメインは，古いホストアドレスを新たな名前空間にもち込むために用いられている．

NIS＋と同様に，DNSの名前はあるオブジェクトを底とするサブドメインの列で表現される．ホスト名とメールボックスが主に用いられるオブジェクトであるが，様々なオブジェクトを名前づけすることができる．システムは拡張可能なので，ユーザは新たな種類のオブジェクトを付け加えることができる．

あるノードのドメイン名は，特定的なものから順に並べたラベルのリストである．ルートは空

のストリングである．名前は内部的には，長さを表すバイトとオクテットの列で表される．0の長さのバイトはルートに達したことを示す．

名前の外部表現は，長さのバイトを除き，その代わりに名前をドットで分離する．最後につけられたドットはルートを表す(すなわち，完全なドメイン名であることを表している)．相対的な名前はドットをもたないものである．ホスト内のソフトウェアが何をつければよいかを知っている．相対名はよく知られたデフォルトの名前に対して相対的であるか，サーチリストから得られる．

完全なドメイン名のオクテットの数(ラベルのオクテットとラベルの長さの総計)は255である．数字で始まる名前などを用いることもできるが，そうしないことが推奨されている．大文字，小文字の区別は保存されるが，比較や検索では意味をもたない．

9.7.2 ゾーン

DNS階層の中にはゾーンという概念がある．ゾーンとは，名前空間の任意の連続した部分である．例えば，最上位のCom.はゾーンである．一連のサーバが，そのゾーンで権威あるものとされる．

そのゾーンの下には，Sun.Com.で参照することのできるSunというゾーンがある．Sun.Com.のサーバはその下に属する，例えば下に挙げるような多くのドメインに関して責任をもつであろう．

Marketing.Sun.Com
Engineering.Sun.Com
Hyperbole.Marketing.Sun.Com
Whitepapers.Marketing.Sun.Com

ゾーンが，Marketingの方にEngineeringの方よりも深く伸びていることに注意してほしい．Engineeringにサブドメインをほしいと思うなら作れない理由はない．例えば下のようなものが考えられる．

FDDI.Engineering.Sun.Com

リゾルバが，名前の問合せに対して権威のある答を必要とする場合にはいつも，名前空間のそのゾーンの情報を扱っているサーバを見い出さなければならない．ゾーンに分割することは，権限を名前空間の下の部分に委譲する方法である．

9.7.3 実装

Sunのネットワークでは，以下の4種類のサーバがある．

- 1次マスタ
- 2次マスタ

・キャッシング専門サーバ

・フォワーディングサーバ

マスタネームサーバは，そのゾーンに関する問合せに対して権威ある回答をもっているものである．マスタネームサーバのうちの1つが1次マスタである．DNSでは，すべてのゾーンが少なくとも2つのマスタサーバをもたなければならない．実際，NICに新たな下位ドメインを登録するには，その組織はそのドメインのための複数のサーバを用意しなければならない．

あるゾーンのマスタサーバのうちの1つが1次マスタである．DNSゾーンファイルの更新はすべて1次マスタで行われる．その他の1つ以上のマスタは2次マスタである．これらはNISのマスタ・スレーブと同様な関係である．

よくある設定では，あるゾーンの1次マスタが，他のゾーンの2次マスタを兼ねる．このようにすれば，ゾーンが複数のマスタをもたなければならないという要請は満足される．NISと同様に，2次マスタは定期的に1次マスタに変更があったかどうかをチェックする．

残りの2種類のサーバは権威をもっていないと見なされる．キャッシング専門サーバは，情報のキャッシュをもっている．いつかはキャッシュは古くなり，権威のあるサーバからリフレッシュされる．

以前解決した問合せを記憶しているという意味において，すべてのサーバはキャッシングサーバである．キャッシング専門サーバは権威のある情報をもっていない特殊な場合である．

4種類目のフォワーディングサーバは，単にすべての問合せを他のサーバに転送する．転送のためにサーバのアドレスのリストをもっている．

特殊な例としてフォワーディングスレイブがある．これは常に問合せを特定のマスタに転送し，そのマスタが問合せに答える．これはローカルネットワークがあって，名前を解決するためにインターネットに1点だけから通信を行おうとする場合にこうなる．なぜこうするのだろうか．真のリゾルバは，ワークステーションでは扱えないほど大きな量のキャッシュをもつことができる．

最後に，単にネームサーバへのアクセスと答の解釈の方法だけを知っているライブラリであるリゾルバがある．すべてのサーバは，他のサーバへのアクセス法としてリゾルバをもっている．クライアントは単にリゾルバのみをもち，namedデーモンを走らせていないマシンのことである．

9.7.4　設定

Sunのワークステーションが起動すると，DNSのリゾルバになる．すべてのリゾルバ(すべてのサーバを含む)は，問い合わせるサーバのアドレスのリストが示されている/etc/resolv.confファイルを用いる．ここでブートストラップのアドレスをもっておくことで，NISの項で触れた問題，すなわちサーバを見い出すためにブロードキャストを行い，したがってサブネットワークごとにサーバを必要とするという問題を解決している．NIS+およびDNSのいずれにおいても，サーバ

```
root# cat /etc/named.boot

;  named.boot file for Primary master

;  type    domain          source file or host
;  First line is the home directory for the name server
;  (this lets subsequent files to have relative path names).
directory  /var/named

;  The way to find the root servers is via a cache file.
cache                              named.ca

;  This server is the primary server for this zone
;  "puhosts" is the name of the zone file for this zone
primary   Podunk.edu    puhosts

;  This line says I'm the primary server for this domain
;  (which is the same as above)
primary   32.128.in-addr.arpa    puhosts.rev

;  This host is the primary server for the local host
;  loopback, data to be found in named.local
primary   0.0.127.in-addr.arpa   named.local
```

9-10 named.bootファイル

を広く分散させることが可能である．ブートストラップファイル中でIPアドレスを用いることで，組織内のどこにサーバを置くかを決める自由度が大きくなる．もちろんInternetの世界では，国中どこにサーバを置くことも可能である．

resolv.confファイルに加えて，namedデーモンの設定のためにnamed.bootファイルが（やはり/etcに）ある．このファイルは，そのサーバが1次になるか，キャッシング専門になるか，フォワーディングになるかを決める．このファイルは，他のサーバやファイルの位置も格納している．

図9-10に，1次サーバ用のnamed. bootファイルの例を示す．この同じファイルに，2次サーバになるための情報を含むこともできる．

```
secondary podunk.edu 128.32.0.4 128.32.0.10 puhosts.zone
```

secondaryというキーワードは，そのノードが2次サーバであることを示す．データはそのゾーンのデータをもっているサーバのアドレスである．ファイル名は，そのファイルにそのゾーンのデータをバックアップとして置いておくことを示している．キャッシング専門サーバのブートファイルは，単にキャッシュの行から構成されるであろう．フォワーディングサーバのブートファイルは，次の例のように，複数のアドレスの集まりである．

```
;named.boot file for Forwarding server
forwarders 128.32.0.10 128.32.0.4
```

```
root# cat named.ca
; Initial cache data of root domain servers
; list of servers:

                9999999    IN    NS    NIC.DDN.MIL.
                9999999    IN    NS    A.ISI.EDU.
                9999999    IN    NS    TERP.UMD.EDU.
                9999999    IN    NS    C.NYSER.NET.
                9999999    IN    NS    AIS,BRK,NUK.
                9999999    IN    NS    GUNTER-ADAM.AF.MIL.
; and their addresses:
NIC.DDN.MIL.    9999999    IN    A     10.0.0.51
NIC.DDN.MIL     9999999    IN    A     26.0.0.73
C.NYSER.NET.    9999999    IN    A     192.33.4.12
AOS.BRL.MIL.    9999999    IN    A     128.20.1.2
AOS.BRL.MIL.    9999999    IN    A     192.5.22.82
NS.NASA.GOV.    9999999    IN    A     128.102.16.10
TERP.UMD.EDU.   9999999    IN    A     10.1.0.17
A.ISI.EDU.      9999999    IN    A     26.3.0.103
GUNTER.AF.MIL.  9999999    IN    A     26.1.0.13
```

9-11 named.caファイル

named.ca ファイルは重要なファイルである．これにはルートサーバの名前が書いてある．この情報は，DNS の機能にとって非常に重要である．どんな名前を解決するにも，階層の頂点から始めて，降りていかなければならない．図 9-11 に，このファイルの例を示す．

9.8 リソースレコード

サーバにとって最も重要なファイルはゾーンのデータである．この情報は，サーバがそのクライアントのために保持しているものである．ゾーンデータは，ある情報をもった一連のリソースレコードによって構成されている．いくつかの標準のリソースレコード形式が定義されている(図 9.12)．また，システムは拡張可能である．

それぞれのリソースレコードは次に挙げる標準フォーマットを用いている．

{名前}　{余命}　クラス　レコードの種別　データ

下に簡単な例を示す．

SRI-NIC.ARPA. 999 IN A 10.0.0.51

このレコードは，(NIC のサーバである) SRI-NIC.ARPA という名前に関するレコードである．このデータは 999 秒有効であり，Internet クラスに属する．この場合のリソースレコードの種類はアドレスレコードであり，データは (論理的に十分であり) Internet プロトコルアドレスである．こ

A	address	ホストのアドレス.
CNAME	Canonical name	nickname から 正しい名前を導くのに用いる
HINFO	Host Information	CPUとオペレーティングシステムの種類
MX	Mail Exchange	与えられた名前宛のメールを受け取るホストの名前とその希望値
NS	Name Server	与えられたゾーンにおける権威をもつホストの名前. 典型的には, リゾルバは次にA型のレコードに記録されている, このホストのアドレスを知ろうとするであろう.
PTR	Pointer	CNAMEに似ているが, IN-ADDR.ARPAのような特殊なドメインで用いられる.
SOA	Start of Authority	ドメインの名前と, そのドメインの責任者と, 設定情報を格納している.
WKS	Well-Known Service	Internet に特有のリソースであり, 特定のホストにより与えられる, よく知られたサービスを示す. IPサービス供給者（すなわちUDPとTCP）のそれぞれに対してビットマップがある. ビットマップは, それぞれのサービスに用いられるポートに対応する場所に, ビットがセットされている.

9-12 リソースレコードに入る情報

の情報を保持することは, リゾルバがあるホストの Internet アドレスを知りたいときに, サーバはそのドメイン名の A レコードを探そうとすることを意味する.

サーバは, ゾーンにリソースレコードにロードし, 問合せに答え始める. ゾーンの先頭には, サーバに関する情報を与える Start of Authority レコードがある. 例を次に示す.

```
@ ( hostmaster.sri-nic.arpa
    45              ; serial
    3600            ; refresh
    600             ; retry
    3600000         ; expire
    86400)          ; minimum
```

@マークは, 現在の起点を示し, この例では明らかに木のルートである. レコードのデータ部分にはホストの名前(SRI-NIC. ARPA)と, そのゾーンの責任者(hostmaster.sri-nic.arpa)が書かれている. 責任者は実際には単に電子メールのアドレスである.

残りのデータはゾーンの設定情報である. serial は, そのゾーンファイルのバージョン番号である. これは2次サーバが, 最新のデータをもっているかを判断するのに用いられる. refresh は2次ネームサーバが更新が必要かどうかを調べるのにどのくらい待つべきかを秒単位で示している.

retry 指標は失敗のあとでどのくらい待つべきかを示す. これはホストに対する要求が多すぎて失敗したときに, 要求が常に繰り返し出されることでホストへの要求が溢れてしまわないようにするためのものである.

最後に, レコードには期限切れと最小値の指標が示されている. 期限切れは, いつデータが古くなったと見なすべきかである. ここでは 3 600 000 秒, すなわち 42 日間である. 最小値は, そのゾーンにおける他のレコードの余命(time-to-live : ttl)の最小値を秒単位で表す. 86 400 は, デフ

ォルトが1日であることを表している．

Start of Authority(SOA)レコードの後に，ゾーンファイルには他の様々なリソースレコードが続いている．さらに，いくつかの特別な文字がある．すでに現在の起点を表す@マークは見た．これは，すべての部分的なドメイン名に付加されるデフォルトとなっている．

起点は，次のようにして設定できる．

```
$ORIGIN engineering.sun.com.
```

ゾーンファイルでは，完全なドメイン名はピリオドで終わるものである．ピリオドで終わっていないドメイン名には現在の起点が付加される．例えば，

```
FDDI
```

は，下のようになる．

```
FDDI.Engineering.Sun.Com.
```

起点を設定することで，小さいグループのユーザは，必要がない限りより大きな組織を意識しないですむ．したがって，特別に他のドメインを指定しない限り，資源も局所的なものを指す．

9.9 DNSの動作

ゾーンファイルは，特定のサーバが，インターネット中のリゾルバに対して提供している情報を定義している．この情報は問合せと回答のプロトコルによって利用できる．各々のDNSの問合せは1つの質問を含み，回答は0,1または複数の答を含む．

図9-13に，DNSメッセージの標準フォーマットを示す．問合せのパケットを受けると，いくつかのビットが反転され，答が後ろに付け加えられて，呼び出したノードに送り返される．1つの問合せに複数の質問を入れることが可能なことに注意してほしい．

しばしばクライアントは，問合せの型によって示される特定の種類の情報しか必要としない(図9-14参照)．また，そのゾーンのすべてのデータのように，大量の情報を必要とする問合せもある．

図9-15に，典型的なMXレコードに関する問合せを示す．ヘッダは，質問は1つであり，Stanford.Eduに関するMXの質問であることを示している．この問合せには，ホストの名前(またはno recursion desiredビットが立っているので，問い合わせるべきネームサーバの名前)が返される．

MXレコードは，それぞれの名前に対して以下の2つの情報をもっている．

・優先度

・ホスト

優先度には，そのホストが得る優先度を示す数が入る．ある特定のユーザが，いくつものMXレコードをもつことができる．もし名前が，いくつかの異なるMXレコードに解決されたときには，クライアントは優先度の値が小さいものから用いる．もし，それが利用可能でないときは，より大きい値のものを用いる．

コード	名前	解説
固定ヘッダ		
ID	Query ID	16ビットのID．照合のため，回答にコピーされる
QR	Query Response Bit	1ビット(0は，問合せ)
OPCODE	Query Type	4ビットの問合せの型
		0: 普通の問合せ
		1: 逆の問合せ
		2: サーバの状態の問合せ
		3-15: 使われていない
AA	Authoritative Answer	1ビット．回答するネームサーバが質問セクションのドメイン名に対して権威をもつサーバである場合にセットされる．回答セクションは別名のために複数の所有者名を含むことがあることに注意せよ．AAビットは，問い合わされた名前か，回答セクションの最初の所有者名に対応する．
TC	Truncation	1ビット．回答の長さが，その通信回線で許されていることを示す．
RD	Recursion Desired Bit	
RA	Recursion Available Bit	
Z	Reserved	3ビット．使われていない．0でなければならない．
RCODE	Response Code	4ビットの回答コード
		0: エラーなし
		1: フォーマットエラー
		2: サーバでの失敗
		3: 名前エラー(問合せの名前が存在しない．この答えは権威あるサーバから送られる)
		4: インプリメントされていない(例えばinverse)
		5: 方針に反するため拒否
		6-15: 将来用
QDCOUNT	Question Count	質問セクションの中のエントリの数(16ビット)
ANCOUNT	Answer Count	回答セクションの中のRRの数
NSCOUNT	NS Count	権威あるレコードセクションの中のネームサーバのリソースレコードの数
ARCOUNT	Additional Resource Count	列挙された追加のRR の数
質問セクション		
QNAME	Questions Name	ドメインの名前(ルートを示す長さ0のオクテットで終わる)．パディングを行わないので奇数オクテットになることもある．
QTYPE	Query Type	2オクテットのコード（すべての型フィールドおよび，＊やAXFのようなその他のコードを含む)
QCLASS	Query Class	インターネット，ワイルドカード，または，他のクラス

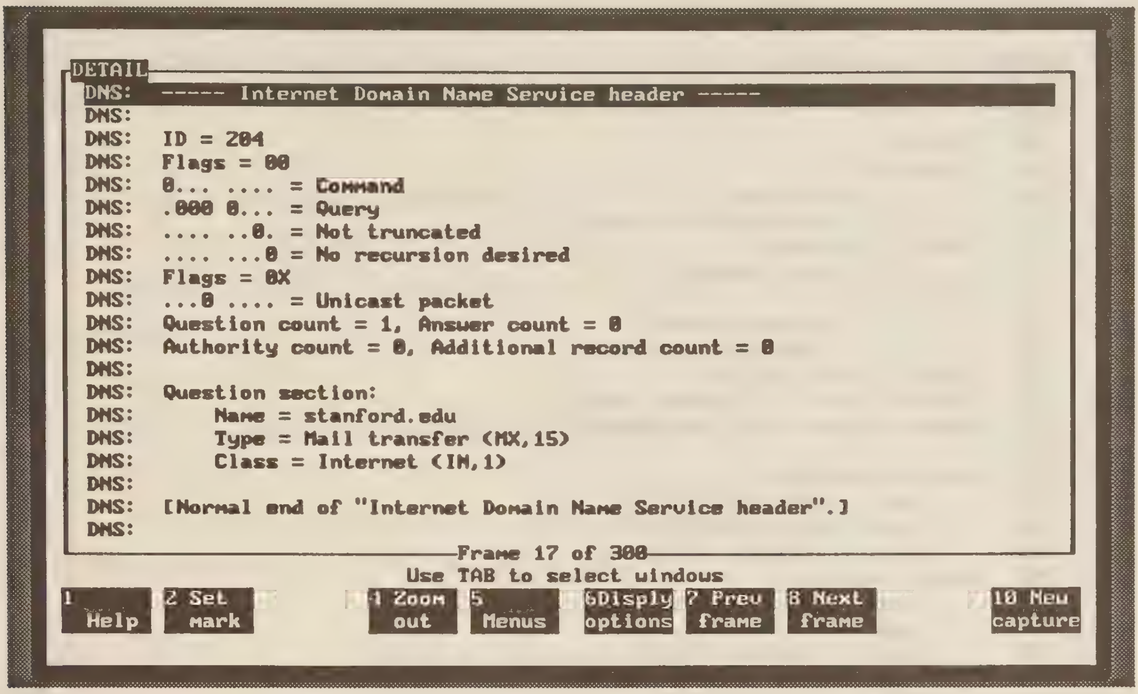
9-13 DNS メッセージフォーマット

回答セクション		他の3つのセクションは同一の形式で，COUNTセクションで示された数のリソースレコードからなる．すべてのRRは，以下のように構成される
NAME		この回答が言及する名前
TYPE		2オクテット の RRの型コード
CLASS		2オクテット
TTL		32ビット 符号なし整数
RDLENGTH		リソースレコードデータセクションの長さ
RDATA		答

9-13（続き）　DNS メッセージフォーマット

問合せの型	数	解説
リソースレコードに基づく問合せ		
A	1	ホストのアドレス
NS	2	権威をもつネームサーバ
MD	3	メールの宛先（現在では用いられていない——MXを用いよ）
MF	4	メールの転送先（現在では用いられていない——MXを用いよ）
CNAME	5	別名に対する正規名
SOA	6	権威ゾーンの始点
MB	7	メールボックスのドメイン名 (実験用)
MG	8	メールボックスのグループメンバ(実験用)
MR	9	メールのドメイン名を付け替える(実験用)
NULL	10	実験用
WKS	11	よく知られたサービスの記述
PTR	12	ドメイン名ポインタ
HINFO	13	ホストの情報
MINFO	14	メールボックスまたはメールリストの情報
MX	15	メールエクスチェンジ
TXT	16	テキスト文字列
複数のリソースレコードに基づく問合せ		
AXFR	252	ゾーン全体の転送要求
MAILB	253	メールボックスに関係するレコード(MB，MG，MR)の要求
MAILA	254	メールエージェントのRRの要求（現在では用いられていない——MXを用いよ)
*	255	すべてのレコードの要求

9-14　DNSの問合せ



Courtesy of Network General

9-15 DNSの問合せ

例えば，ユーザ Carl Malamud が下のような MX フィールドをもっていたとしよう.

Carl.Malamud.Com MX 0 UUnet.Com

Carl.Malamud.Com MX 10 OUTHost.Com

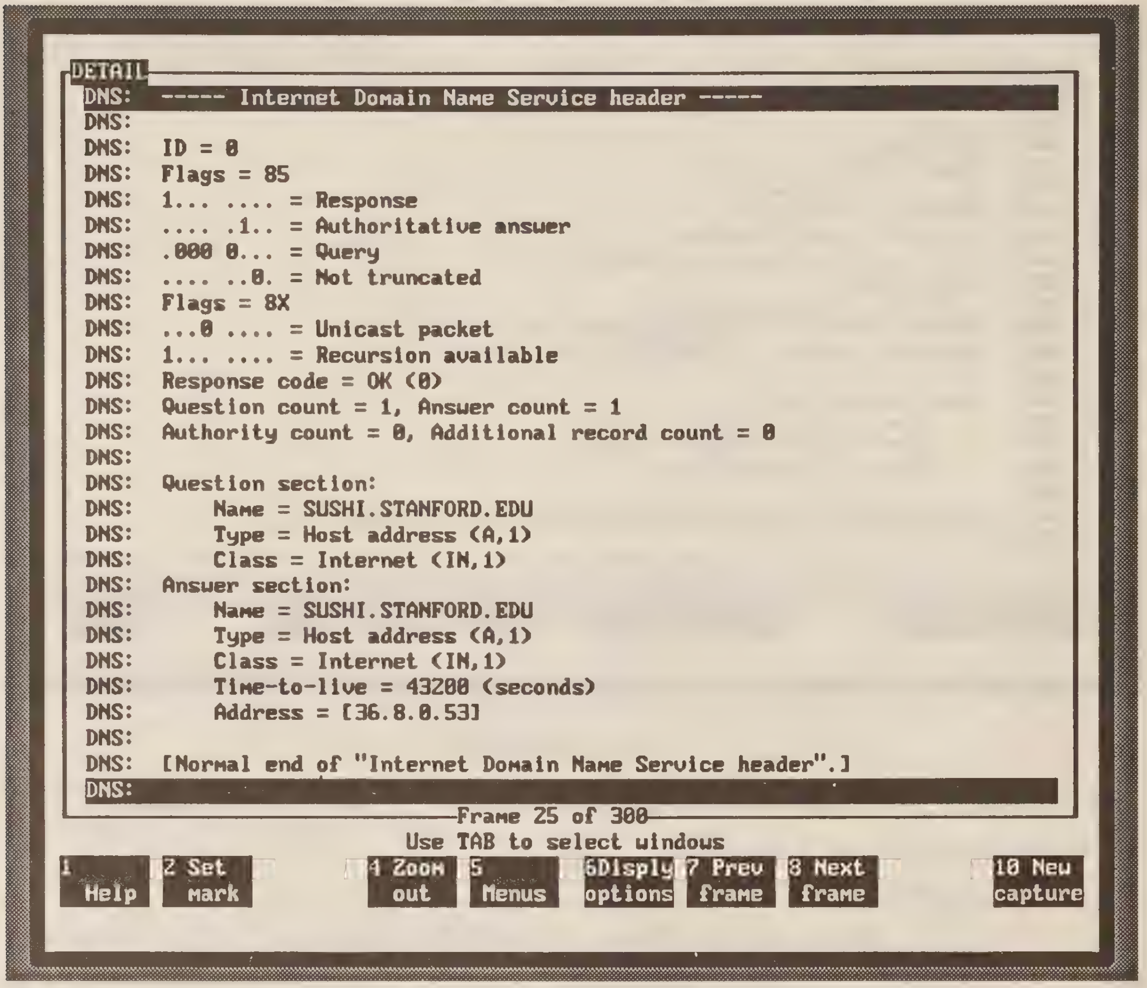
このレコードは，Carl@Malamud.Com 宛にメッセージを送るには(ゾーンファイルにデータを格納するときには@マークをピリオドに置き換えることに注意してほしい)，まず UUnet.Com を試してみることを意味する．これは必ずしも Carl@Malamud.Com が，UUnet にメールボックスをもっていることを意味しない．ホスト UUnet が，このメッセージをどうすればよいかを知っていることを意味するだけである.

あるドメインに多くのユーザがいることはあり得る．MX レコードに，アスタリスクを付けることで，そのサブドメインすべてにマッチするようにすることができる.

*.Malamud.Com MX 0 UUnet.Com

このレコードは，Malamud.Com のすべてのサブドメインが，UUnet.Com によって取り扱われることを示している.

別の問合せに対する回答を図 9-16 に示す．このパケットでは，回答ビットと，権威のある回答であることを示すビットがセットされている．回答セクションにはホスト Sushi.Stanford.Edu のインターネットアドレスが含まれている．このホストに対する TTL はかなり長く 12 時間である．こちら側のコンピュータ内のリゾルバは，同じ質問を繰り返さずにこの情報を 12 時間キャッシュ



Courtesy of Network General

9-16 DNSの答

しておくことができる.

9.9.1 IN-ADDR.ARPA ドメイン

ホストやメールボックスを見い出すことも DNS の重要な機能であるが，もう 1 つ重要な機能がある．それはネットワークやネットワークを扱うゲートウェイを見い出すことである．これは動的な経路設定プロトコルではなく，静的なブートストラップ機能である．

基本的な考え方は，ある IP ネットワークのアドレスが与えられたときに，そのネットワークに関して責任をもつゲートウェイを見つけることである．逆アドレス（Inverse Address）と呼ばれる人工的なサブドメインを作成し，ARPA ドメインに置くことによってこの問題は扱われている．インターネットアドレスを IN-ADDR.ARPA ドメインに付けることにより，ネットワークの IP アドレスを DNS のシンタックスによって表現することができた．

単一のホストの場合のやさしい例から見てみよう．ホストの IP アドレスは，アドレス空間の 32 ビットすべてが使われていることを除いて，ネットワークアドレスに似ている．IP アドレスの 4 つの数字を反転させて(したがって DNS が求めるように左から右に)，下に示すリソースレコードを作る．

51.0.0.10.IN-ADDR.ARPA PTR SRI-NIC.ARPA.

73.0.0.26.IN-ADDR.ARPA PTR SRI-NIC.ARPA.

ホスト SRI-NIC.ARPA. は，以下の 2 つのインターネットアドレスをもっている．

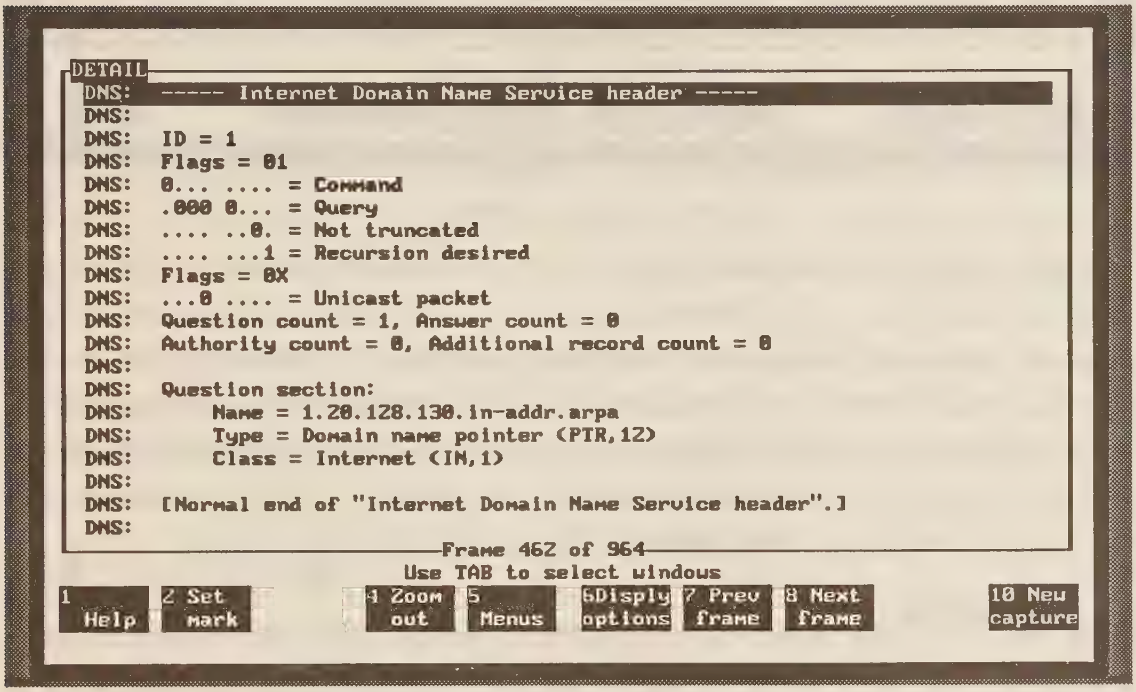
10.0.0.51

26.0.0.73

この IP アドレスをもつポートをオープンする依頼を受けたら，DNS の問合せを発行して，ホストの名前を知ることができる．

ネットワークに関しては，単に IP アドレス空間の一部分を用いればよい．

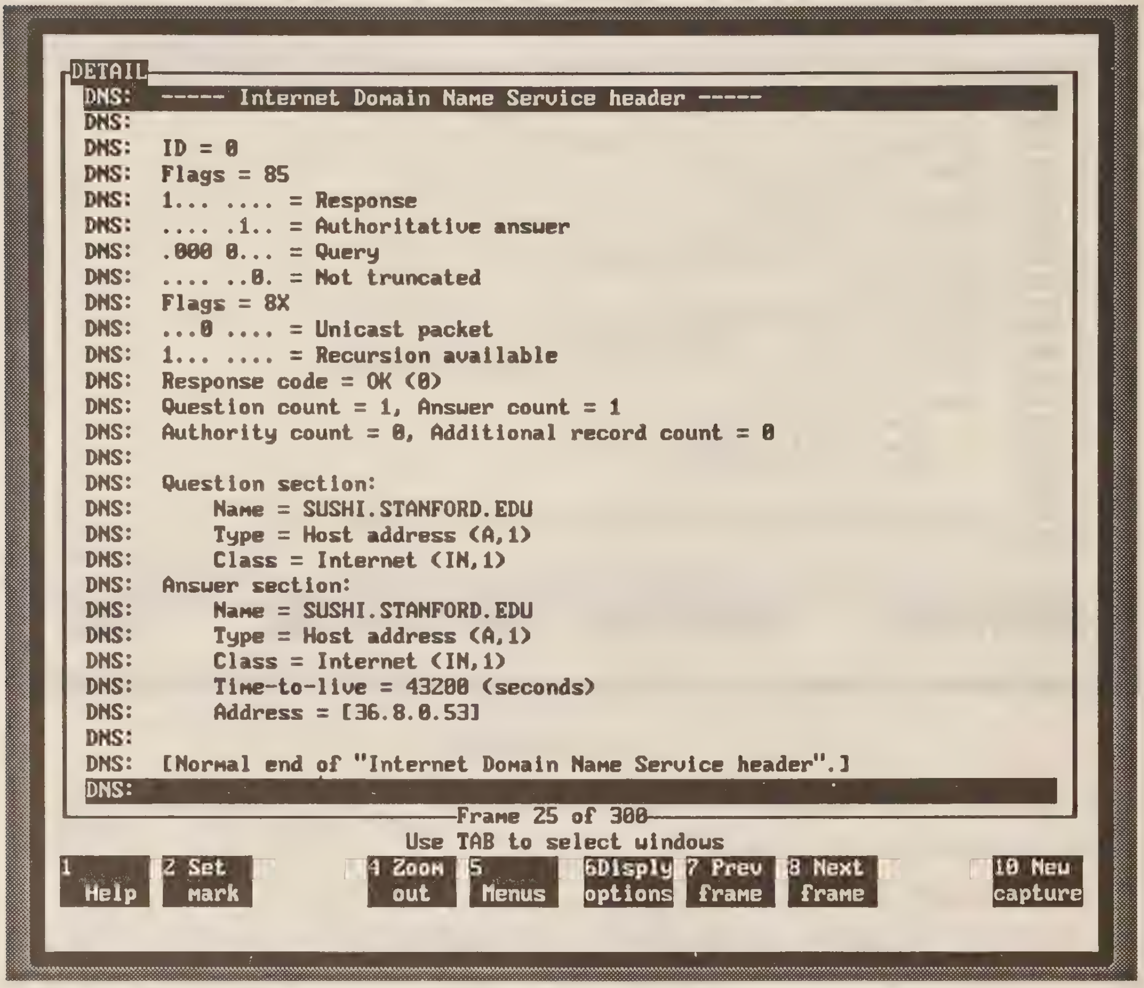
10.IN-ADDR.ARPA	PTR GW.CSL.SRI.COM.
18.128.✣	PTR GW.CSL.SRI.COM.
33.12.192.✣	PTR GW.CSL.SRI.COM.



Courtesy of Network General

9-17 in-addr.ARPAの問合せ

✣(訳注) IN-ADDR.ARPA が後についているべきである．



Courtesy of Network General

9-16 DNSの答

しておくことができる.

9.9.1 IN-ADDR.ARPA ドメイン

ホストやメールボックスを見い出すこともDNSの重要な機能であるが，もう1つ重要な機能がある．それはネットワークやネットワークを扱うゲートウェイを見い出すことである．これは動的な経路設定プロトコルではなく，静的なブートストラップ機能である．

基本的な考え方は，あるIPネットワークのアドレスが与えられたときに，そのネットワークに関して責任をもつゲートウェイを見つけることである．逆アドレス（Inverse Address）と呼ばれる人工的なサブドメインを作成し，ARPAドメインに置くことによってこの問題は扱われている．インターネットアドレスをIN-ADDR.ARPAドメインに付けることにより，ネットワークのIPアドレスをDNSのシンタックスによって表現することができた．

単一のホストの場合のやさしい例から見てみよう．ホストの IP アドレスは，アドレス空間の 32 ビットすべてが使われていることを除いて，ネットワークアドレスに似ている．IP アドレスの 4 つの数字を反転させて(したがって DNS が求めるように左から右に)，下に示すリソースレコードを作る．

51.0.0.10.IN-ADDR.ARPA PTR SRI-NIC.ARPA.

73.0.0.26.IN-ADDR.ARPA PTR SRI-NIC.ARPA.

ホスト SRI-NIC.ARPA. は，以下の 2 つのインターネットアドレスをもっている．

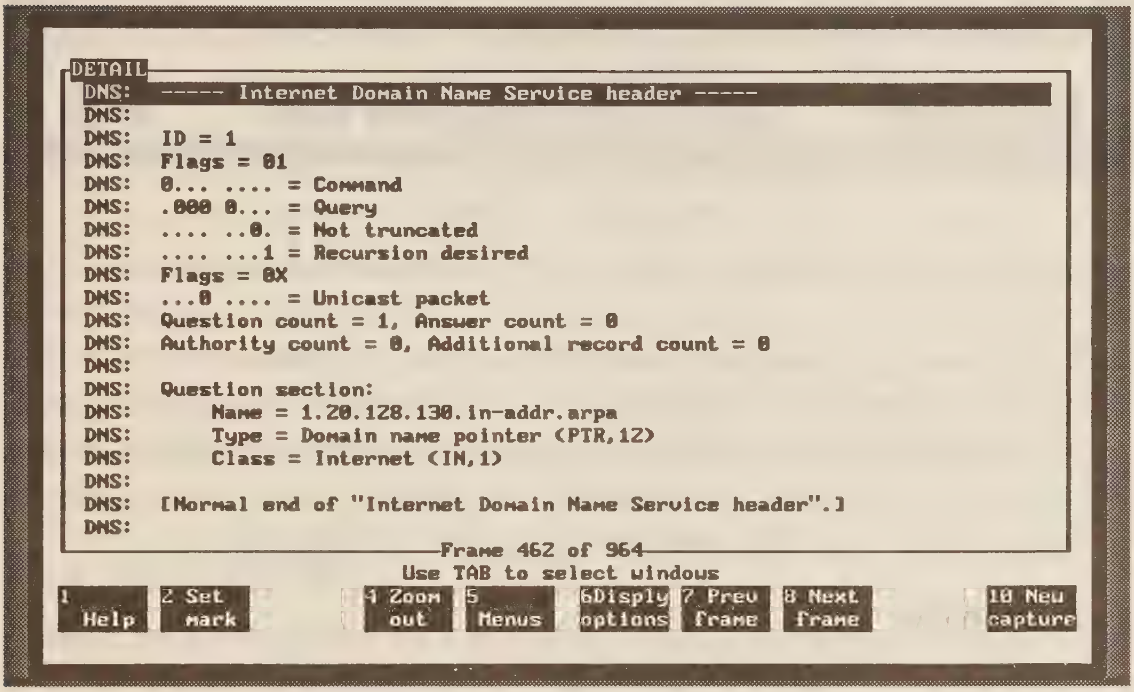
10.0.0.51

26.0.0.73

この IP アドレスをもつポートをオープンする依頼を受けたら，DNS の問合せを発行して，ホストの名前を知ることができる．

ネットワークに関しては，単に IP アドレス空間の一部分を用いればよい．

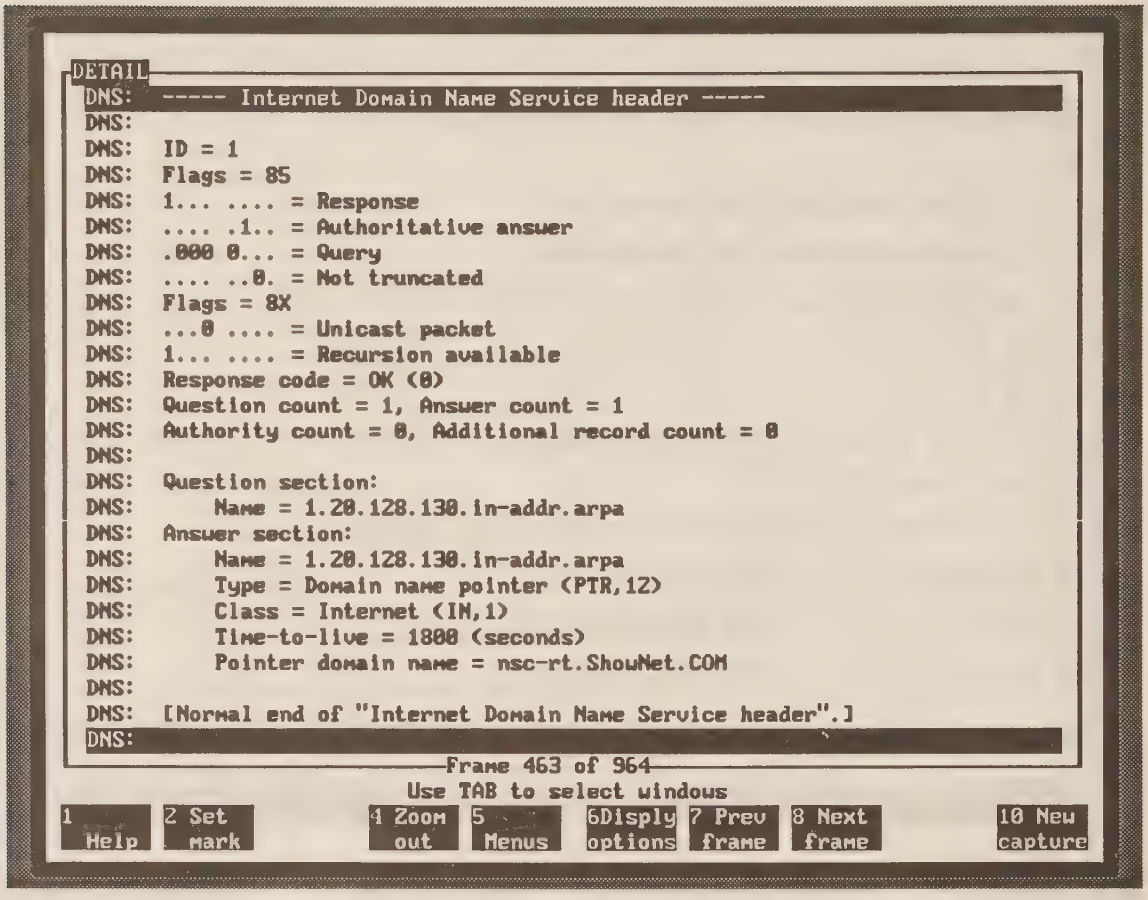
10.IN-ADDR.ARPA	PTR GW.CSL.SRI.COM.
18.128.✣	PTR GW.CSL.SRI.COM.
33.12.192.✣	PTR GW.CSL.SRI.COM.



Courtesy of Network General

9-17　in-addr.ARPAの問合せ

✣(訳注)　IN-ADDR.ARPA が後についているべきである．



```
DETAIL
DNS: ----- Internet Domain Name Service header -----
DNS:
DNS:  ID = 1
DNS:  Flags = 85
DNS:  1... .... = Response
DNS:  .... .1.. = Authoritative answer
DNS:  .000 0... = Query
DNS:  .... ..0. = Not truncated
DNS:  Flags = 8X
DNS:  ...0 .... = Unicast packet
DNS:  1... .... = Recursion available
DNS:  Response code = OK (0)
DNS:  Question count = 1, Answer count = 1
DNS:  Authority count = 0, Additional record count = 0
DNS:
DNS:  Question section:
DNS:      Name = 1.20.128.130.in-addr.arpa
DNS:  Answer section:
DNS:      Name = 1.20.128.130.in-addr.arpa
DNS:      Type = Domain name pointer (PTR,12)
DNS:      Class = Internet (IN,1)
DNS:      Time-to-live = 1800 (seconds)
DNS:      Pointer domain name = nsc-rt.ShowNet.COM
DNS:
DNS:  [Normal end of "Internet Domain Name Service header".]
DNS:
                     Frame 463 of 964
                  Use TAB to select windows
1 Help  2 Set mark  4 Zoom out  5 Menus  6Disply options  7 Prev frame  8 Next frame  10 New capture
```

Courtesy of Network General

9-18　PTRレコードの応答

10,128.18 または 192.12.33 で始まるすべてのアドレスは GW. CSL. SRI. COM. という名のゲートウェイに送られる．

図 9-17，図 9-18 に，この種類の操作の例を示す．図 9-17 に示される問合せは，インターネットアドレス 130.128.20.1(アドレスが反転されていることに注意)をもつホストの名前を問い合わせている回答から，そのホストの名前は nsc-rt.ShowNet.Com であり，見本市会場にあるホストのアドレスであることがわかる．見本市会場は変化しやすいネットワークとして知られているので，TTL が 12 時間ではなく 30 分と極端に短くなっていることに注意してほしい．

9.9.2　再帰問合せ

再帰問合せでは，サーバがリゾルバになり，他のサーバに問合せを行う．これによって，クライアントは実際の答がどこにあるのかを知らなくてすむ．別の言葉でいえば，仕事がクライアントからサーバに移行することになる．

再帰問合せを用いるには，クライアントとサーバの双方がそれをサポートしていなければなら

ない．再帰問合せを用いる同意は，問合せおよび回答メッセージ中の次に示す2ビットを使って交渉される．

・Recursion Available(RA)

・Recursion Desired(RD)

再帰問合せを提供可能なサーバは，常にRAビットを立てる．クライアントはそれを見て，RDビットを立てた問合せを出す．クライアントがRDビットを立てたとしても，そのサーバから以前にRAビットの立ったメッセージを受け取ったときでなければ，それに依存することはできない．実際に再帰問合せが行われたかどうかがわかるように，回答にはRAビットとRDビットの両方がセットされる．

もちろん，サーバは特に要請されない限り再帰問合せを行わない，ということが重要である．これは無限ループを防ぐためである．名前の構成がループになっていたとき(例えば，あるサーバは他のサーバが権威をもつといい，そのサーバは元のサーバが権威をもつという場合)，自動的に再帰問合せが行われるとループから抜ける方法がない．

9.9.3 キャッシュ

リゾルバは，ネットワークの遅延や渋滞，サーバの過負荷を防ぐために，キャッシュをおおいに利用しようとする．どんなキャッシュでも問題になるのが，データが古くなっているかもしれないということである．権威をもつサーバはどのリゾルバがキャッシュされたデータをもっているかわからないので，ユーザにデータの変更を知らせようとはしない．その代わりに，TTL(time-to-live)が，そのデータがどのくらいの期間用いられるべきかを制御する．

しばしば，リゾルバはキャッシュされたデータを用いようとする．そのデータがエラーを引き起こした場合，リゾルバは直接権威をもつサーバに接触し，最新の情報を引き出す．しばしば，間違った情報を使おうとする過程によって，キャッシュが破棄される．

リゾルバはオプションとして，否定的な結果をTTLだけキャッシュすることができる．典型的な例は，ある名前が存在しなかったときである．これは権威あるサーバが答えるときに，回答にSOA RRを加えることで実現される．そのときには，SOA中のTTLの最小値が失敗のキャッシュに用いられる．

9.9.4 メッセージの圧縮

DNSにおけるラベルは，長さの指示子と実際のデータによって表現される．とても長いドメイン名を何度も何度も何度も参照しなければならない場合，どうすればよいだろうか．ドメイン名の最後にある一連のラベルは，ポインタで置き換えることができる．ポインタは，最初の2ビットがセットされた2オクテットの列で表される．

最初の2ビットをセットすることで，ポインタとラベルを区別することができる(ラベルの長さ

は63オクテットまでであるから，ラベルの最初の2ビットは0である)．オフセットはメッセージの先頭(ドメインヘッダのIDフィールドの最初のオクテット)からのものである．これによりメッセージの中に次の3種類のドメイン名が存在することになる．

・0のオクテットで終わる一連のラベル

・ポインタ

・ポインタで終わる一連のラベル

もちろん，自ホスト内のリゾルバはユーザに結果を示す前に，これらの3種類の名前を統一する．

9.10 名前サービスの組合せ

これまでに，NIS，NIS＋，DNSの3つの名前サービスを見てきたが，他にも重要な名前サービスが少なくとも2つある．X.500とOSF/DEC Naming Serviceである．

X.500は，DNSと同じ種類の問題を，より完璧な(したがってより複雑な)方法で扱っている．X.500のディレクトリは，属性値と型で探索することができる．また，はるかに多様な階層的な名前づけ様式をサポートしている．

X.500は，電子メール環境でよく用いられる．例えば，PSI商用ネットワークやその他の環境で稼働しているWhite Pagesプロトタイプがその例である．X.500は，局所的な環境において高速な参照を行うことを期待されていない．

DEC Naming ServiceはDECnet Phase Vアーキテクチャの一部であり，OSF(Open Software Foundation)にも採用されている．NIS＋もDECのシステムも，単一の組織の中に限られた名前解決サービスを提供している．どちらも階層的な性質をもっているため，会社全体にわたる名前空間の管理に適している．

多くの識者が，これらのサービスは互いに補い合うべきものと見ている．今日ではDNSが，将来的にはX.500が，各組織のネットワークの端にあるメールハブ(mail hub)や，ルーティングゲートウェイなど適切なマシンへの道筋を与える糊としての役割を果たす．

いったんある環境に着けば，局所的な名前サービスがとって代わる．局所的な名前サービスは装置や組織の境界の中に限られているので，それほど一般的である必要はない．これによってNIS＋の表参照のようなより洗練された動作が可能になる．

ここで理解すべき重要な点は問題が2つあることである．1つは名前の管理である．これはすべての組織が行わなければならないことであり，2つ目の問題である名前空間を実現するソフトウェアよりも，はるかに難しい問題である．

より進んだ勉強のために

Malamud, *Analyzing DECnet/OSI Phase V*, Van Nostrand Reinhold (New York, 1991). DNA 名前づけサービスについて述べられている.

P. Mockapetris, *DNS Encoding of Network Names and Other Types*, RFC 1101, April, 1989

———, *Domain Names - Concepts and Facilities*, RFC 1034, November, 1987.

———, *Domain Names - Implementation and Specification*, RFC 1035, November, 1987.

Sun Microsystems, *System & Network Administration*, Part Number: 800-3805-10, Revision A of 27 March, 1990. NFS, NIS その他一般的な管理の問題について述べられている.

Chapter 10

セキュリティ

10.1 安全なネットワーク?

Internetのアプリケーションの大半は，充分なセキュリティを備えていない．ユーザはTelnetやFTPのようなサービスを利用する際にパスワードを入力するが，そのパスワードは平文のままパケットに格納されて相手に送られるだけである．これは，Internetのアプリケーションの多くは，そもそも遠隔の計算機資源へのアクセスをしやすくするためにつくられたものであって，アクセスを制限するためのものではないからである．ただし，このことはUnixシステムおよびTCP/IPネットワークを安全にするのは不可能だということを意味するのではない．光ファイバをはじめとして，経路選択における制御，セションレベルでの認証，そして2つのアプリケーションがやりとりするメッセージの暗号化に至る様々な手段を選択して使うことができる．もちろん，本書ではこれらの方法をひとつひとつ詳細に見ていくことはせず，まずあまり安全でない場合について検討してから，次により強力なセキュリティ機構のうちの2つについて調べていくことにする．

パスワードを平文のままネットワークに流す方式は，受動的な手段を用いた攻撃に非常に弱い．なぜなら，ユーザはネットワークを容易に傍受できるからである．大半のワークステーションでは，イーサネットアダプタを無差別モード(promiscuous mode)にして，ネットワーク上のすべての（あるいは大半の）パケットが手に入るようにすることができる．そして手に入れたパケットを調べて最初のログインのためのパケットを探し出し，このパケットからパスワードを取り出すという動作をするユーティリティを作るのは容易である．

平文のままのパスワードが許されるのは，次の2つの場合である．

- 信頼のおける環境
- セキュリティを気にする必要のない環境

信頼のおける環境とは，例えばローカルな仕事場である．その場所およびネットワークが無認可のアクセスから物理的に隔離されていれば，平文のパスワードを用いても大した問題にはならない．また，アクセスされるのが誰でも利用可能な文書サーバならば，目標は破壊を防ぐことであ

って，アクセスを制限することではない.

パスワードが解決するのは，セキュリティの問題の一部 (離れた相手の認証(authentication)) だけである. これは，あるユーザ名を入力した人が実際にそのユーザ名に対応する人なのかどうかを決める原始的な方法であるが，もしもユーザがパスワードを忘れてしまったら，あるいは誰かがパスワードを推定してしまったら，もちろんこの認証機構は破れてしまう.

ユーザの認証が済んだら，その次に来るのが，その人がしてよいことを決める認可(authorization)の問題である. システムの中には，このアクセス制御をまったく行わないものもある. 例えばドメイン名前システム(Domain Name System : DNS)では，すべてのユーザがあらゆるデータに対してアクセスすることができる. しかし，DNS がアクセス制御を行わない(その意味で認可を行わない) からといって，このサービスが何の役にも立たないということにはならない. 単に秘密の情報を DNS 上に置いておかなければいいだけの話である.

最後に可視性(accountability)の問題がある. あるユーザが認証され，あるレベルの認可が得られた場合，その人が認可されていないファイルにアクセスしようとして 1 日を過ごしているのかどうかわかった方がいいであろう. そのシステムで人々が何をしているかを (もちろん個々のプライバシーを尊重しつつ) 知ることは，各ユーザに対する監視を可能にする.

これらの基本的なセキュリティを実現するためのツールには様々なものがあるが，ここで重要なことは，セキュリティは 1 ヶ所だけのものではないということである. 1 つのネットワークの周囲には何重もの防壁が張り巡らされ，それによって複数のレベルでの保護が提供され，いくつもの検問所が設けられる. 例えば，RPC のレベルでは幅広い範囲のセキュリティ方策が実現されている. これは一連の防壁であり，侵入者はこれを乗り越えなくてはならない. これに加えて，たいていのオペレーティングシステムでは，カーネルのまわりにもう 1 つ防壁を巡らせている. RPC サーバがユーザに何を許そうと，カーネルはそれとは無関係にどの資源を使わせるかを判定するのである.

この章では，まず大半の Unix および Internet のサービスがもっている単純な，そして安全でないセキュリティ機構について見ていく. 次いで，いくつかの基本的な RPC のセキュリティ機構 (いずれも異なるレベルの保護を実現する) を説明する.

セキュリティが高いレベルの保護にまで及んでくると，そのための手段は，一般に公開鍵暗号系と対称鍵暗号系の 2 つの暗号系に基づくものとなる. この 2 つの暗号系およびそれらと Sun のセキュリティとの関係についても述べる.

10.2 単純なセキュリティ機構

パスワードを平文のままネットワークに流すやり方は安全でないだけでなく，面倒でもある. ユーザは何回も何回もパスワードを打ち込むのに疲れ，やがてパスワードを埋め込んだスクリプ

トファイルを書くようになる.

この状況について少し考えてみよう. パスワードを入力するのを面倒がり, かつもの覚えの悪いユーザがいるとする. そこで彼は自分のワークステーションのキーボードの下側にパスワードを書いた紙を貼り付け, さらに自分のホームディレクトリに node1.login, node2.login, node3.login といった Unix のスクリプトを作成したとする.

この場合, セキュリティに関して少なくとも 3 つの問題がある. 第 1 に, そのワークステーションが物理的に保護されていないと, 誰でもそのワークステーションの所へ行ってリブートできる. 第 2 に, 抜け目のないハッカーがいれば, キーボードの下を調べてパスワードを見つけるだろう. 第 3 に, そして最悪なことに, いったんハッカーがそのワークステーションのホームアカウントへのログインに成功すると, スクリプトファイルがあるために他の 3 つの計算機にもアクセスできてしまう.

Unix では, hosts データベースによってパスワードのもつ上のような問題点の一部を解決している. Unix システムでは/etc/hosts.equiv および.rhosts の 2 つのファイルにより, 遠隔のユーザがファイルにアクセスできるようになっている. 多数のアカウントが多数の計算機に分散している場合, これは悪くないやり方である. それらのアカウントに到達するには, 単にそのアカウントに“rlogin”すればよく, パスワードさえ不要である.

このセキュリティ機構はルーズなものであるが, 必ずしも悪いわけではない. パスワードをネットワークに流したりせず, 入口のホスト (outermost host) がユーザの認証を行っていて, 防火壁としての役割を果たしている. ここで入口のホストとは, 例えば Internet ゲートウェイ, ダイヤルインシステム, またはワークステーションなどの, その仕事場の中でユーザが最初にログインするホストのことである. ネットワーク上のすべてのホストが信頼できる認証を行うのであれば, その仕事場の内部でのネットワークセキュリティは多少緩めてもよい. しかし一方, hosts.equiv データベースに登録されたホストを偽装したホストを用いて, サービス要求を捏造できるという問題がある.

このような方式を用いる場合には, 入口のホストにログインするときに用いるパスワードをできるだけ安全にすることが必要である. この安全性を高めるのに役立つツールがいくつか存在する. 例えば, Sun ではパスワードを定期的に変更させたり, 単語の奇妙な組合せを提供してパスワードの推定をより困難にしたりすることにより, セキュリティを向上させることができる.

10.3 RPC とセキュリティ

RPC はアプリケーションに対して基本的な認証サービスを提供する. 現状では以下の 4 種類の方式が定められている (将来これらに追加していくことも容易にできる).

- 無認証 (none) 型

・Unix 型
・Secure(DES)RPC
・Kerberos RPC

無認証型はその名の通り認証を行わない．Unix 型は Unix システムが用いているものに似た原理に基づくもので，無認証型よりも多少は安全性が高い．Secure RPC は公開鍵暗号系と DES 対称鍵暗号系を組み合わせたものである．4 番目の Kerberos RPC は，MIT が開発した Kerberos 認証サービスを適用したものである．

無認証型について説明しても仕方がないので Unix 型から始めることにしよう．アプリケーションが Unix 型の認証を利用するためには，2 つの手続きが必要となる．すなわち，クライアントの側では信任状(credential)を作成する手続きが，サーバの側ではその信任状を利用するための手続きがなくてはならない．

Unix 型では，信任状はかなり簡単に作ることができる．クライアントは，自分のコードに RPC ライブラリの authunix_create_default への手続き呼出しを挿入するだけでよい．これによって信任状一式が作成される．この信任状には呼出し側のユーザ ID(UID)，グループ ID(GID)，その他の所属グループのリストおよびクライアントのホスト名が含まれている．

クライアントは次に，これらの信任状を取り出して"クッキー(cookie)"と呼ばれる隠蔽された(opaque)データ構造体に納める．すると以後のすべての RPC 呼出しにこのクッキーが含まれ，Unix 型の認証を行うサーバの側で，手続きを呼び出して信任状を取り出し，それを調べることができる．その後サーバが，そのユーザの身分に基づいてどのようなアクセス権を与えるのか，それはアプリケーション次第である．

しかし認証は，信任状と確認証(verifier)の 2 つがあって初めて完全である．悪意のある侵入者は信任状一式を偽造することができるが，Unix 型ではそれが本物であるかどうかを確認する方法がない．侵入者がスーパーユーザなら，これは特に容易である．つまり，自分が偽装したいユーザに身分を切り替えるだけでよい．

NFS では，マウント要求に含まれている送信者の Internet アドレスを調べて信任状中のホスト名フィールドの確認証とし，さらに特権ポートから来た要求しか扱わないことによって，この問題に対処しようとしている．この場合でも，偽のフィールドを捏造して裏をかくことは可能であるが，しかしこれには相当の決意と労力が必要である．

以上のように，Unix 型の認証は信頼のおける環境に適したものである．この機構を破ることは可能であるが，信頼できる環境下ではあまり重大な問題ではない．

Unix 型には，あまり強力ではないことの他にも問題があり，それはこの Unix 型という名前が表している．MVS, DOS そして VMS といったオペレーティングシステムは，すべて異なる方式を用いていて，これらのセキュリティ機構の相互変換を Unix 型を中心に確立するのは容易ではない．しかし繰り返すが，信頼のおける環境下での複数の Unix ワークステーションという構成の場

合なら，Unix 型は効果的な認証機構である．

10.4 Secure RPC

Secure RPC は Unix 型よりずっと強力な認証機構である．この方式は次の 2 種類の暗号系に基づく．

・対称鍵暗号系

・公開鍵暗号系

対称鍵暗号系では，2 人以上のユーザがメッセージを暗号化したり復号したりするのに使う鍵を共有する．Data Encryption Standard (DES) はこの対称鍵暗号系の代表例である．“対称”という言葉は送信者と受信者が同じ鍵を共有するところからきている．

対称鍵暗号系における問題は，どのようにして同じ鍵を複数のユーザにもたせるかという点である．後述する Kerberos（対称鍵暗号系のみを用いて認証を行う）では，信頼のおける鍵配送センターを設けることによってこの問題を解決しているが，もう 1 つの解決策として，別の暗号系である公開鍵暗号系を用いて鍵を暗号化し，それを各ユーザに配送するという方法がある．この公開鍵暗号系では，ユーザは各自秘密鍵 (private key) と公開鍵 (public key) の 2 つをもつ．個々の秘密鍵は安全なところに保管され，決して明らかにされない．一方，公開鍵は必要ならば誰でも手に入れることができる．

この 2 つの鍵は，互いに逆の機能をもつ．秘密鍵で暗号化された情報は，公開鍵を使わない限り復号することはできない．またその逆も成り立つ．そこで，あるユーザからのものと称するメッセージを受け取った場合，単にそのメッセージを復号してみるだけでよい．もしうまく復号できれば，その暗号化されたメッセージは秘密鍵をもっている送り主しか作れないから，受け取ったメッセージが本当にそのユーザからのものだとわかるのである．

以上で公開鍵暗号系についての世界一短い講義は終わったので，今度はこの暗号系が Secure RPC においてどのように機能しているかを見ていくことにしよう．Secure RPC では，まずすべてのユーザがネットワーク全体に通用するユーザ名をもつ．これは次のような形式である．

unix.uid@NIS_subdomain.NIS_domain

ここで NIS_subdomain.NIS_domain とは，NIS+に納められているネットワーク上のグループ名である．この形式がうまく働くためには，各ユーザが一意の UID をもっていなくてはならず，そのため名前づけサービスが必要である．

ユーザ名の中に“unix”のラベルがあるため，より幅広い名前づけができる．例えば Secure RPC を VMS でも通用させたいときには，次のように VMS 用の名前に置き換えればよい．

VMS.<uid, gid>@subdomain.domain

ユーザ名にはホスト名が示されておらず，ユーザはネットワーク上のすべての計算機を（その

人のアクセス権に応じて）使用することができる．しかしUnixにおけるスーパーユーザの場合は，次のようにどの計算機でのスーパーユーザなのかを明示しなくてはならない．

unix.nodename@NIS_subdomain.NIS_domain

これで認証のための準備はできた．ユーザはネットワーク全体に通用する一意なユーザ名をもち，ある計算機にその計算機のログイン機構によってログインしている．さらにユーザは自分の秘密鍵および公開鍵をもっていて，公開鍵は(おそらくNIS＋テーブルの属性の1つとして)公開されている．

一方，秘密鍵はそのユーザのファイルシステムの中に，ユーザのパスワードで暗号化されて保管されている．秘密鍵がきちんと隠蔽されているかどうかにこのセキュリティ機構全体の成否がかかっているので，この秘密鍵の暗号化は特に重要である．

Secure RPCでは，認証手続きを開始するのはクライアントである．ユーザはまずDESで用いる鍵を任意に(あるいはライブラリに選んでもらって)決める．この鍵は会話鍵(conversation key)と呼ばれ，そのクライアントとサーバの一連の交信の間だけ使用される．会話鍵はユーザの秘密鍵によって暗号化される．クライアントがサーバに送る信任状は，この暗号化された会話鍵の他に，クライアント名とウィンドウを含んでいる．このウィンドウは信任状の有効期限を示すもので，この期限を越えたタイムスタンプのついているメッセージは受取りを拒否される．ウィンドウは会話鍵で暗号化されている．

クライアントは，サーバに信任状だけでなく確認証も送る．この確認証は，誰か他の人が以前に送られた信任状を記録し，過去の交信を真似ようとしているのではないということを示すためのものである．確認証は2つの情報からなる．1つは現在の時刻であり，もう1つはその確認証のウィンドウである．両者とも会話鍵で暗号化されている．

サーバは信任状から会話鍵を取り出すことができる．サーバが情報をクライアントに返すときは信任状は必要なく，確認証のみを送る．この確認証は，クライアントが送ってきたタイムスタンプから1を引いた(そしてもちろん会話鍵で暗号化した)ものである．タイムスタンプから1を引いて送り返すのは，相手が本物のサーバかどうか，サーバを偽装している何かではないのかどうかをクライアントが確認できるようにするためである．

これ以後のこのクライアントとサーバの間の情報交換においても，この手順が繰り返される．クライアントは自分のID（もう信任状全部を送る必要はない）と現在の時刻を会話鍵で暗号化した確認証を送り，サーバは送られてきた時刻から1を引いたもので応答する．

以上で，信任状を作成してサーバに送る方法が確立された．サーバのRPCライブラリはこの信任状が正式なものかどうか（例えば正しく復号できるかどうか）を検査することができる．しかし，RPCライブラリは検査結果などに基づいて，アプリケーションに対して何をしろと指令することはできない．行動を決めるのはアプリケーションである．後述するSecure NFSは，アプリケーションの動作をどのように決めるかを示す1つの事例となるだろう．

10.4.1 Diffie-Hellman 法

Secure RPC は公開鍵暗号法の 1 つである Diffie-Hellman 法を用いている．この Diffie-Hellman 法，およびもう 1 つの有名な公開鍵暗号法である RSA 暗号法は，いずれもその公開鍵と秘密鍵が，互いに数学的に逆数であるということに基づく．具体的には，2 つの鍵は非常に大きな数の因数である．このとき 2 つの鍵のうち 1 つの鍵を用いて暗号化し，もう 1 つの鍵を用いて復号するというのは容易であるが，一方の鍵から他方の鍵を見つけるのは，大きな数の因数分解が必要となるため，計算量的に実行困難である．

Diffie-Hellman 法では，秘密鍵も公開鍵も 192bit の数で，公開鍵はあるよく知られた定数の，秘密鍵を指数とする累乗である．したがって，あるユーザ A の公開鍵は，ある定数の，A の秘密鍵を指数とする累乗に等しい．

あるホストとクライアントの間の交信において相手方のユーザとなる B も，自分の秘密鍵と公開鍵をもっている．このとき A と B との間で使う共通の鍵は，B の公開鍵を A の秘密鍵を指数として累乗をとることによって導くことができる．また，同じ鍵は，A の公開鍵を B の秘密鍵を指数として累乗をとっても得られる．

A は自分の秘密鍵を知っており，また B の公開鍵は容易に手に入るので，共通の鍵を得ることができる．同様に，B も自分の秘密鍵と A の公開鍵がわかっているので，共通の鍵が導ける．しかし，A と B 以外にこの 2 つの情報を 2 つとももっている人はいないので，共通の鍵は外部には秘密のままである．

上の 2 つの式が同じ鍵を A と B に与えるのはなぜだろうか．公開鍵はある定数の秘密鍵を指数とする累乗であることを思い出してほしい．したがって，共通の鍵は，定数の，一方の秘密鍵を指数とする累乗の，さらに他方の秘密鍵を指数とする累乗となる．

ここで A の公開鍵は定数の A の秘密鍵を指数とする累乗であるから，この共通の鍵を与える式を少し変形して，“定数の，A の秘密鍵を指数とする累乗”を“A の公開鍵”に置き換えることができる．また，この逆も成り立つ．すなわち，共通の鍵は，A の公開鍵の，B の秘密鍵を指数とした累乗をとることでも導けることになる✣．

✣(訳注)　式で表すと次のようになる．

ユーザ A の公開鍵と秘密鍵をそれぞれ P_A, S_A，ユーザ B の鍵を同様に P_B，S_B とすると，ある定数（本文中の“定数”）C に対して P_A と P_B は次のように定められる．

$$P_A \doteq C^{S_A} \pmod q$$
$$P_B \equiv C^{S_B} \pmod q$$

ただし q は 2^{192} に近い（そして 2^{192} より小さい）素数である．

ここで，A と B との共通の鍵を $K\,(0 \leqq K < q)$ とすると，次式が成立する．

$$K \equiv {P_B}^{S_A} \equiv C^{S_A S_B} \equiv {P_A}^{S_B} \pmod q$$

したがって A と B は共通の鍵 K を得ることができる．またそれ以外のユーザは（S_A も S_B も知らないので）K を算出することはできない．以上は Diffie と Hellman の論文（IEEE Trans. Info. Theory, Vol.26, No.6, pp.644-654）による．

10.4.2 スーパーユーザの NFS へのアクセス

通常, 1 つの計算機においてスーパーユーザであるからといって, 他のサーバにおいても同じ特権が与えられるわけではない. あるクライアントのスーパーユーザがリモートファイル階層をマウントしても, それに対するスーパーユーザとしてのアクセスは拒否されるのが普通である (ただし注目すべき例外として, ディスクレスワークステーションの場合がある).

サーバに対するアクセスに関しては, スーパーユーザはユーザ ID 0 を使うことはできず, 代わりに別のユーザ ID 番号を割り当てられる. 普通は, その UID は 65 534 であり, これはユーザ名 nobody に対応する. つまり, クライアントにスーパーユーザとしてログインしているユーザが, リモートファイルシステムに対してもつアクセス権は, 一般の部外者に与えられるアクセス権と同じ, すなわちたいていの場合まったく無権利である.

しかし NFS では, 指定されたクライアントにおけるスーパーユーザが, サーバにおいてもスーパーユーザとしてアクセスできるようにすることができる. そのためには, 次のようにそのクライアントのホスト名を/etc/exports のリストに書き込めばよい.

```
/usr/src -root=samba
```

また次のようにすれば, 1 つ以上のホスト名を指定できる.

```
/usr/src -root=samba:tango:mambo
```

セキュリティをまったく考慮しないのなら, すべてのクライアントプロセスがスーパーユーザの UID を与えられて, つまりスーパーユーザとしてアクセスできるようにすることも可能である.

```
/usr/src -anon=0
```

このコマンドはユーザに信任状なしでのアクセスを許し, かつその際にスーパーユーザの UID を与えるものである. したがって, 誰でもこのファイルシステムに対してあらゆる操作ができる.

anon オプションのより一般的な例は, 次のようなユーザ名 nobody に相当する UID である-2 を与えるものである✣.

```
/usr/src -anon=-2
```

特権ポートを用いると, もう少しセキュリティを向上させることができる. 多くのオペレーティングシステムでは, TCP および UDP はスーパーユーザの特権をもつユーザだけが利用できるポートをいくつか用意している. カーネルを変更すれば, 到来する要求の送信元のポート番号を NFS が調べられるので, それを利用するとスーパーユーザの要求は必ず特権ポートから来なくてはならないようにできる. このようにすれば, ユーザが Unix 型の信任状を捏造してスーパーユーザとしてアクセスしてしまうのを防げる. しかしながら, これはあまり有力な手段ではない.

✣(訳注) SunOS 4.1 からは, ユーザ nobody の UID は 65 534 とするのが正しい.

10.4.3 Secure NFS

ファイルをより強力に保護するため，NFSの代わりにその一種であるSecure NFSを使うことができる．Secure NFSはSecure RPCを土台として構築されていて，したがって各ユーザは公開鍵と秘密鍵をもっている必要がある．さらに，ファイル階層はsecureオプションをつけてエクスポートされなくてはならず，またmountコマンドの方でもSecure NFSによるマウントであることを指示する必要がある．

Secure NFS（とその土台となるSecure RPC）は，公開鍵NISマップを利用している．このマップは次の3つのフィールドをもつ．

```
netname public_key:private_key
```

netnameは各ユーザに固有のネットワーク名である(例：unix.uid@domain)．そしてpublic_keyとprivate_keyはそれぞれ公開鍵と秘密鍵を示すデータである．

こう述べると，鋭い読者は次のように考えるであろう．アクセス制御を行わない名前づけサービスによって公布される秘密鍵の，いったいどこが秘密なのか？　その答はこうである．つまりNISでは，秘密鍵はそのユーザのパスワードで暗号化されているのである．NIS＋でも秘密鍵は暗号化されているが，この情報にアクセスできる人を制限するため，さらに認証とアクセス制御が設けられている．

鍵の初期設定はそれほど難しくない．通常は，誰かスーパーユーザの特権をもつ人に，newkeyコマンドによって鍵を作ってもらう．このnewkeyコマンドは，ユーザ名とホスト名(ホスト名も認証される必要がある）を要求する．もし公開鍵マップに“nobody”というエントリがあれば，ユーザはchkeyコマンドを用いて自分で鍵を作ることができる．

鍵を作成した後はユーザ自身でそれを変更できる．鍵を変更する際には，次のように自分のパスワードを入力しなければならない．

```
user_mach%chkey
Generating new key for malamud
Password:
Sending key change request to NFS Server
Done.
user_mach%
```

もしセキュリティが重大な問題なら(Secure NFSを使うぐらいなのだから当然重大問題であるが)，ネットワーク中に通用する名前空間には“nobody”のエントリはない方がよい．もしあると，誰でも新しいエントリを公開鍵データベースに作れてしまう．

他のすべてのNISマップと同様，各エントリは必ずしもすぐには伝播しない．マップは即座に更新されるかもしれないし，さもなければユーザは（更新情報の伝播が翌朝までに完了するのを

期待しながら）その夜は帰宅を余儀なくされるであろう．

ユーザがログインするとき，ログインプロセスは publickey.byname NIS マップからそのユーザの秘密鍵を取り出す．そしてパスワードを用いて鍵を復号し，それを記憶しておく．

しかし，計算機にログインする際にパスワードを使わないこともある．ユーザが遠隔のホストから rlogin コマンドを用い，そしてホストの hosts.equiv ファイルにそのホストの名前が入っている場合がこれにあたる．この場合，Secure RPC を使うためには，keylogin コマンドを実行してパスワードを入力しなければならない．これによって秘密鍵を復号し，keystore ファイル✣ に納めることができる．

ただし，rlogin コマンドを用いてホストにログインした上で，認証の必要なアプリケーションにおいて Secure RPC を用いるというのは少々馬鹿げている．なぜなら，このホストは安全であるべきだと定めたというのに，そのホストのユーザの認証は遠隔のホストに任せてしまっているのだから．これは安全なローカルネットワークの中でのみ使えるやり方である．

秘密鍵はユーザがログインしている間だけ/etc/keystore ファイルに保管されるが，keylogout コマンドを用いてログアウトする前に秘密鍵を消去してしまうこともできる．

鍵の準備ができたら，NFS におけるアクセス制御の方針を決めなくてはならない．具体的には，secure オプションによって Secure RPC を認証のために使うよう指示し，access コマンドによってファイルに到達できるクライアントを指定する．例えば：

```
/usr/src -secure,access=engineering
```

同様に，クライアントの側では secure マウントをファイルシステムテーブルまたは Automounter マップ中に指定する．ファイルシステムが secure オプションでエクスポートされたのに，ユーザの側で mount コマンドに secure オプションを付け忘れたらどうなるだろうか？　この場合，ユーザはリモートファイルにアクセスできることはできるが，ただしすべてのアクセスはユーザ nobody のものとして行われる．したがって，誰にでも開放されているファイルしか読むことができない．

10.5 Kerberos

Kerberos は，Athena プロジェクトの一部として，MIT によって開発された認証サービスである．Kerberos とは，具体的には信頼のおける第三者的な認証サーバで，ユーザの身分を他のコンピュータに証言してくれるものである．

Kerberos のクライアントは多様で，ユーザや様々なサーバもこれに含まれる．これらのクライアントのために，Kerberosはデータを暗号化したり復号したりするのに使う鍵のデータベースを

✣（訳注） SunOS 4.1 からは，秘密鍵はプロセス keyserv の中に（つまりメモリ中に）保管される．

管理している.

クライアントも Kerberos に保管された自分用の鍵を知っているので，Kerberos とそのクライアントは共通の，そして外部には秘密の鍵をもち，これを用いて両者の間のメッセージを暗号化することができる．これが Kerberos の 2 つの鍵のうちの 1 つである．Kerberos のクライアント (ユーザおよびサーバ) は，この共通の鍵を用いて，Kerberos に自分を認証させることができる.

次のステップは，Kerberos のクライアントのうちの 2 つが互いに通信できるようにすることである．これには交信鍵(session key)を用いる．Kerberos が交信鍵を用意して，それを 2 つのクライアントに伝えるのである．これにより，この 2 つのクライアントは共通の鍵をもつことになり，互いに通信できるようになる.

すべてのアプリケーションが，同じ Kerberos の使い方をするわけではない．クライアントが誰であるか知る必要のないサービスもあるし，もう少し安全性を要するアプリケーションで，交信の始めに認証を行わなくてはならないものもあるだろう．それよりもずっとセキュリティに配慮して，ひとつひとつのメッセージに対していちいち認証を必要とするようなサーバもあり得るし，また真の安全性を達成するため(ただし非常に遅いであろうが)，メッセージ自体がこの共通の鍵で暗号化されるようにすることもできる.

鍵はデータの暗号化のために用いられる．暗号化の方法は様々であるが，たいていのシステムでは Data Encryption Standard (DES)の方式のうちの 1 つを採用している．しかしどの暗号法を使うかは，Kerberos においてはあまり重要ではない．Kerberos が関係してくるのは，鍵を各クライアントに配る方法についてである.

Kerberos のデータベースでは，1 つのクライアントごとに 1 つのレコードをもっている．このレコードには，クライアント名，鍵，そして鍵の有効期限が納められている．人名や，電話番号などの情報は，Hesiod や NIS＋のような名前サービスに保管される．Hesiod とはドメイン名前システムを土台とした解放型サービスで，DNS にアクセス権とユーザの認証のための機構を付加したものである (Hesiod も MIT の Athena プロジェクトの一部であるが，Sun では提供されていない).

Kerberos のある環境で，あるサービスの安全性を確立するための基本的な方法は，ユーザがサーバに次の 2 つの情報を送ることである.

- 信任状 (Kerberos ではチケット(ticket)と呼ばれる)
- チケットが盗まれたものではないことを示す確認証
  (Kerberos では認証状(authenticator)と呼ばれる)

ネットワークを通じてサービスが使用できるようになるまでには，次の 3 段階を経る．まず，ユーザは Kerberos との間に信任状を確立する必要がある．次に，ユーザは Kerberos に依頼して，あるサービスに対して自分を認証してもらう．そして最後に，ユーザは新しい信任状をそのサーバに送る.

とりあえずKerberosサーバに対する認証は済んでいるものとして，交信チケットがどのような内容のものであるかを見ることにしよう．交信チケットは，クライアントにあるサーバへのアクセスを許すものであるが，その内容は次の通りである．

・サーバ名
・クライアント名
・クライアントのIPアドレス
・現在の時刻
・チケットの有効期限
・Kerberosが任意に選んだ交信鍵

ここで与えられる交信鍵は，このサーバとクライアントとの間の，この特定の交信においてのみ有効である．チケット全体はサーバの鍵によって暗号化されている．サーバ以外でサーバの鍵をもっているのはKerberosだけであるから，このようにすることにより，クライアントがチケットをサーバに送ったとき，サーバはこのチケットがKerberosからクライアントに渡されたものだとわかるのである．

なぜチケットをサーバの鍵で暗号化するのだろうか？　これはクライアントがチケットをいじり回して，例えば有効期限やIPアドレスを変えてしまうのを防ぐためである．

チケットも認証のための認証状の一種である．しかしいったん認証されたら，今度は過去の交信を真似ている何か別のものに，そのクライアントが取って代わられてしまっていないかどうかを確認しなくてはならない．このための確認証（または認証状）はクライアント名，タイムスタンプ，そしてクライアントのIPアドレスの3つの情報からなり，交信鍵によって暗号化されている．

チケットは，チケット交付サーバ(ticket-granting server)と呼ばれる特殊なサーバから与えられる．このサーバは，クライアントに他のサーバと通信するためのチケットを提供する役目をもつが，これには古典的なブートストラップ問題がつきまとう．すなわち，始めにチケット交付サーバと通信するためのチケットはいったいどうやって手に入れたらいいのか？

ワークステーションからログインしようとするとき，ワークステーションはユーザ名を入力するよう求めてくる．しかし，ワークステーションはそれを自分のUnixファイルと照合するのではなく，このクライアントがチケット交付サービスを利用してもいいのかどうかを照会するパケットをKerberosに送る．

Kerberosは，自分がそのクライアントを知っているかどうか調べて，もし知っていれば，チケット交付サーバとの通信を許可する特別なチケットを作成する．このチケット交付チケットは，まもなくクライアントが通信してくるので準備するよう通知するために，まずチケット交付サーバに送られる．

次に，Kerberosはクライアントにそのユーザのパスワード（実際はそのパスワードを引数とす

る関数の値)で暗号化したパケットを送る．このパケットの中身は次の2つの情報からなる．1つは，チケット交付サーバと通信するためのチケット（前にチケット交付サービスに送られたものと同じ)であり，もう1つはこのクライアントとチケット交付サーバとの間で使う交信鍵である．

この時点で，ワークステーションは2つの情報をもつ．その1つはユーザ名で，もう1つは暗号化されたパケットである．そこでワークステーションはユーザにパスワードを入力するよう促し，このパスワードを用いてパケットを復号し，チケット交付チケットと，認証状を暗号化するための鍵を取り出す．

ユーザのパスワードはネットワークを流れたり，ワークステーションの中に格納されたままになったりしていないことに注意してほしい．ワークステーションは，戻ってきた最初のパケットを復号するためだけにパスワードを用いるのである．これ以後のすべての通信は，チケット交付サーバとの通信用の交信鍵を用いて行われる．

そのうち，チケット交付チケットは期限切れになるが，このときユーザは再びパスワードを入力するよう求められる．しかし，どんなに悪賢い盗聴者でも，できることは任意に選ばれた交信鍵を盗むことだけで，これとてもきわめて困難である．

ここまででクライアントは自分の身分を，少なくともチケットの有効時間の間だけ証明できるようになり，したがって所望のサービスを要求することが可能となる．クライアントがチケットを手に入れるには，チケットを要求するパケットを，チケット交付サーバとの間で使う特別な鍵を用いて暗号化したものを送る．

この暗号化されたパケットの内容は，次の3つである．1つ目は，通信したいサーバの名前，2つ目はチケット交付チケットのコピー，そして3つ目は認証状，つまりクライアント名，IPアドレス，そして現在の時刻の3つを暗号化したものである．

これに対し，チケット交付サーバは2つの情報を返す．1つはアプリケーションサーバへ送るチケットで，もう1つは暗号化に用いる鍵である．いうまでもなく，これらはすべてクライアントとチケット交付サーバとの間の交信鍵で暗号化されている．

これでクライアントは所望のサービスを要求できるようになった．クライアントはチケット(サーバの秘密の鍵で暗号化されている）と，交信鍵で暗号化した認証状を送る．サーバは受け取ったチケットを復号して，これが有効なものであるかどうか確認する．

サーバにはもう1つやることがある．それは認証状を検査することである．サーバは認証状を復号して，時間が有効であるかどうか確かめる．このためクライアントとサーバの時計は当然合っていなくてはならず，さもないとサーバはこのチケットは再送されたものではないかと疑うであろう．

かなり疑り深いサーバは，受け取ったタイムスタンプを順に見て，それが単調に増加しているかどうか確かめることができる．これは自分の相手が本当にそのクライアントであるということのかなり強力な証拠となる．疑り深くなりたがるのは何もサーバだけではない．クライアントも

サーバに身分を証明しろと要求することができる．

このサーバからの応答の検査を行うときは，サーバは受け取ったタイムスタンプに 1 を加える．これは Secure RPC の場合と非常によく似たやり方である．1 を加えた後，このタイムスタンプは交信鍵で暗号化され，送り返される．

10.5.1　Kerberos と RPC

Kerberos を RPC に適用するのは，(少なくとも概念的には)それほど困難ではない．Sun では，AUTH_KERB と呼ばれる新しい認証機構が追加されている．この機構では，Kerberos のチケットの検査は RPC ライブラリが行うため，鍵配送センター(key distribution center : KDC)とのやり取りは，RPC を利用するアプリケーションから見て透明である．

NFS で利用する場合は，export と mount の両コマンドに，新しい Kerberos のためのオプションを付ける．どの認証機構が使われているかがクライアントにわからなくても，mount プロトコルがその情報を返してくるので，この手順はユーザにとって透明なものとなっている．

Secure NFS と同様，Kerberos の利用者は次の 3 つのうちのどれかである．

- ユーザクライアント：username.instance
- スーパーユーザクライアント：root.hostname
- サーバ：nfs.servername

ユーザクライアントの instance はなくてもよい．

どの認証サービスでもそうだが，遠隔のスーパーユーザクライアントが認証されたからといって，そのクライアントにあらゆるサービスを提供するというアクセス制御の方針が取られているとは限らない．ユーザ“nobody”が相変わらず割り当てられる場合もある．

10.5.2　Kerberos で解決できるのか？

完璧なセキュリティシステムは存在しない．Secure RPC は潜在的に問題点をいくつかもっているが，それは Kerberos も同様である．Secure RPC と Kerberos は，交信を行うのに対称鍵を用いていることなど多くの点において同じである．両者が異なるのは，この対称鍵を配送する方法だけで，Secure RPC では対称鍵を配るのに公開鍵を用いているのに対し，Kerberos では鍵配送センター（Kerberos センター）およびチケット交付サーバが使われている．

10.6　セキュリティの管理

セキュリティの問題の多くは，パスワードの管理がいい加減だったり，様々な問題に対して提示されてきた改善策を実行しなかったりすることから生じている．

問題の大半はパスワードと関係がある．例えば，自明ファイル転送プロトコル(TFTP)を使え

ば，パスワードファイルを盗むことが可能である．Unix ではパスワードファイルは暗号化されているが，少なくともここからユーザ名のリストを手に入れることはできる．たとえ TFTP の使用がローカルな環境内に制限されていても，そのグループ内ならパスワードファイルを盗めることに変わりはない．

一度パスワードファイルを手に入れれば，それを調べて UID が 0（スーパーユーザ）のアカウントやパスワードのないアカウントを探したり，他のパスワードファイルのある場所を見つけたりすることができる．また，パスワードのないアカウントや，パスワードの簡単に見つかるアカウントを直接探索することも可能である．この場合，まず finger プロトコルによってアカウント名を手に入れてから，簡単なパスワードを試していく．

パスワードファイルの問題の簡単な解決策の 1 つは，あるドメインの外からアクセスするときには，内部のとは別のパスワードファイルを使うようにすることである．ドメインの内部では，NIS マップをパスワードとして用いる．そしてゲートウェイのようなドメインの境界にある計算機がそのネットワークへの入口として機能する．このマシンは独自のパスワードファイルをもっていて，FTP のような処理のときにはこれを使うようにする．

ローカルなネットワークの中でのアクセス（おそらく信頼のおける人からのものであろう）も .rhosts および /etc/hosts.equiv ファイルを用いることにより管理されている．この 2 つのファイルはどちらも信頼できるマシンを指定するものであるから，このドメインの外にあるホスト名が書かれているときには，それで本当にいいのかどうかを確かめなくてはならない．また，この 2 つのファイルが誰にでも書込み可能となっていないことも確かめた方がいい．さもないと任意のホストを，簡単に信頼できるホストの列に加えることができてしまう．

セキュリティの問題の大半は開放したままのアカウントがあるために発生する．これは，ベンダが出荷したマシンに，デフォルトの特権付きアカウントがあって，しかもユーザがそのアカウントのパスワードを変えていない場合といった単なる管理不行届である場合もある．VMS はこのデフォルトの特権付きアカウントをいくつかもっていて，そのためこの問題に対して特に弱い．

他にも解決の難しい問題がある．anonymous FTP は，誰でもアーカイブにアクセスできるようにするための慣行であるが，それによると guest と呼ばれるパスワードのないアカウントを用意して，それにより誰でも資源にアクセスできるようにすることになっている✣．確かにこの Anonymous FTP は非常に有用な Internet のサービスであるが，もしこれを提供しているのであれば，各ファイルの保護属性やコンフィギュレーションやその他の情報が適切に保護されているかどうかを確かめなくてはならない．またこの際別のパスワードファイルを用いるのもきわめて重要である．

侵入者はある計算機へのログインに成功したとき，しばしばいくつかの決まった手順を実行に

✣（訳注）　ftp または anonymous という名前のログインできないアカウント（/etc/passwd ファイルのパスワードフィールドが '*' のアカウント）を用意するやり方の方が一般的である．ftpd(8) 参照．

移そうとするが，これらの大半は簡単に防御することができる．よくある手の1つは，プログラムを後で実行させるための予定表を管理するプロセスである cron および at を調べるというものである．

もしも後で実行させるプログラムの使うファイルが誰にでも書込み可能になっていると，別のユーザが予定された仕事の内容を簡単に変えてしまうことができる．この仕事はそれを発行した人（スーパーユーザであることさえあり得る）の権限で行われるため，侵入者は再びこの計算機にログインできるようにしたり，なお悪いことにデータをいじったりすることができてしまう．仕事の予定表を含めて，cron や at が参照するすべてのファイルやプログラムを誰にでも書込み可能にしないことが，これを防止する鍵となる．

その計算機の正規のユーザが知らないうちにプログラムを実行してしまうように仕向けるというのも，侵入者の大半がつけ込んでくるセキュリティホールである．例えば，多くのユーザは自分用のディレクトリをサーチパスの最初の要素に置いている．この場合，もし ls と称するプログラムをそのディレクトリに置くことができれば，そのプログラムはユーザがディレクトリ中のファイルのリストを見ようとするときに常に実行されることになる．

このトロイの木馬版の ls は，何かしら作業（例えば新しいアカウントの追加）をした後で，実際にファイルのリストを表示するだろう．そのため ls を実行したユーザは異常に気づかない．トロイの木馬はすべてのユーザに対して危険であるが，スーパーユーザに対しては特に脅威となる．したがってスーパーユーザは，su, login, Telnet などの大きな被害を及ぼしかねないプログラムに関して，出所のわかっている正規のものを使っているかどうか確かめなくてはならない．

多くの場合，トロイの木馬は後で使えるようにある種の情報を隠しておこうとする．例えば，トロイの木馬版の login プログラムなどは，ユーザのパスワードを手に入れてそれをわからない場所へ隠すであろう．よくある隠し場所は/tmp ディレクトリや，“.. ”（ピリオド，ピリオド，スペース，スペース）または“....”（ピリオド，ピリオド，ピリオド，ピリオド）のような奇妙な名前のファイルの中であったり，あるいは.mail のようなまったく普通の名前のファイルの中であることさえある．

10.6.1 遠隔のホストのブートと Secure NFS

ある夜，深夜に電源故障が発生したとしよう．このとき格納されていたすべての秘密鍵は消えてなくなってしまい，どのプロセスも Secure NFS ファイル階層にアクセスできなくなってしまう．しかしもしスーパーユーザが近くにいれば，彼または彼女がスーパーユーザの秘密鍵を復号するための鍵であるパスワードを入力してくれて，処理を続行することができる．

もう1つの問題は，シングルユーザブートである．普通，Unix のワークステーションでは，ユーザがシステムをシングルユーザモードでブートして，そのユーザにスーパーユーザの特権を与えることができるようになっている．なぜか？ 信じられないかもしれないが，スーパーユーザ

がパスワードを忘れてしまったために，システム管理のための特権がすべて使えなくなってしまうということがあるのである．

Sun システムでは，シングルユーザブートはコンソールのような物理的に保護されたところでしか行えない．しかしその計算機がワークステーションだと，コンソールというのは単なるキーボードである．すなわち，もし誰かを1時間自分のオフィスに1人にしておいたら，その人にはそこのワークステーションをシングルユーザモードでリブートして，ファイルをいくつも書き換えて，それからワークステーションを通常動作に戻すだけの時間があることになる．

この問題の解決策は2つある．1つのやり方として，SunOS の C2 セキュリティオプションを作動させることができる．このセキュリティ機能は，このオペレーティングシステムに標準装備されていて，すべてのブートに対してパスワードを要求するようにし，さらに重要な操作の記録などの機能を追加するものである．

もう1つの解決策は，コンソールの特権を取り消して，ログインするのにパスワードが必要となるようにすることである．しかしもしそうしたら，これは C2 セキュリティ機能の場合でも同じであるが，スーパーユーザはパスワードを忘れることは許されない．パスワードを忘れたら，そのシステムは基本的に使用不可能となってしまう．

ブートにおける問題点の最後は，ディスクレスマシンの場合である．ディスクレスマシンのブートを完全に保護する方法はない．悪い（あるいは“極悪な”といってもいい）ユーザはブートサーバを偽装して非正規のカーネルをブートさせ，そのマシンのユーザの秘密鍵を記録させることができる．もちろんこの場合における問題は，Secure RPC はカーネルと鍵サーバが起動した後で初めて保護を提供するという点にある．それまでは遠隔のブートサーバを認証する方法はない．

これらの点は本当に問題なのであろうか？　実際の環境ではあまり問題ではない．例えば，非正規のブートサーバを作り上げるのは非常に困難か，あるいは不可能である．しかも，もしそれほど問題であるなら，ローカルなディスクからブートするようにするだけで充分なのである．

セキュリティの問題の大半は簡単に解決できる．それらは費用がかかるだけである．Secure RPC を付加すると計算機はわずかに遅くなり，そのためコストが増えることになる．またブートのためにローカルなディスクを設置するには費用がかかる．結局，これはどの程度のセキュリティがその計算機環境にふさわしいかという問題に帰着する．

セキュリティのための世界で最も優れたハードウェアやソフトウェアを備えているとしても，最大の防御線はやはり管理者である．管理者は，経験や記録の丹念な調査およびそのほかの検査器具によって，侵入の防止や検出に多大の効果をもたらすことができる．

10.6.2 本当に安全なのか？

Secure RPC や Secure NFS がセッション開始時の認証において効果を発揮するのは確かであるが，いくつかの弱点もある．第1に，Kerberos やその他のユーザがログインすることを求める

すべてのシステムにおいては，鎖の弱い輪はログインするときのパスワードである．もし誰かがあるユーザのパスワードを当ててしまったら，その人は認証システムに関する限り，そのユーザなのである．

しかし，より攻撃的な侵入は，Secure NFS によって防ぐことが可能である．Secure NFS では認証が（ユーザのパスワードが安全であるという前提のもとに）提供されており，また盗聴のような攻撃も，交信鍵が時間と関係づけられていて，しかも信任状中にウィンドウが設定されているため役に立たない．

サーバ宛のすべてのメッセージを傍受したあと，サーバに接続されているプラグを引き抜くこともできる．このときサーバの信任状テーブルは消去されてしまうので，記録したメッセージを再送して攻撃を加えることが可能である．この場合のただ1つの問題は，ワークステーションは5ないし10分のウィンドウを指定しており，またサーバがリブートするにはそれ以上の時間がかかるという点である．もちろん，たいていのネットワーク管理者は自分のサーバを安全な場所に設置して，個別のワークステーションやユーザによる危険から隔離している．

Secure RPC の機構は，ユーザの公開鍵の離散対数(digital logarithm)をとることによって破ることができるが，しかしこのためには大型のスーパーコンピュータを長時間走らせるといったかなり大規模な道具立てが必要である．つまり，Secure RPC は何気なくトラブルを起こしてやろうと思った人物に攻撃を断念させるには非常に有効であるが，国際的スパイを必死で捜査している国家安全保障委員会の行動を止めることはできないだろう．

もしセキュリティが真に重大問題ならば，アクセス制御と認証を備えた Secure NFS を他の方策と併せて使うようにすべきである．例えば，暗号化と光ファイバを用いたイーサネット型のLAN が存在する．これを Secure NFS と組み合わせ，しかもこのネットワークを物理的に安全な領域内に限定すれば，きわめて高いレベルのセキュリティを実現することができる．

10.6.3　性能

他の機能と同じく，セキュリティもまた計算機資源を消費するため費用がかかる．セキュリティにおけるコストの大部分は暗号の処理のためのものである．DES を用いた暗号化はかなり速い．例えば，タイムスタンプは 64bit 長（たまたま DES のブロックサイズと同じ）であるが，最初の認証が終わった後は，1回の RPC ごとにこのタイムスタンプに対し以下の4つの暗号処理が行われる．

・クライアントが要求メッセージのタイムスタンプを暗号化

・サーバがそれを復号

・サーバが応答メッセージのタイムスタンプを暗号化

・クライアントがそれを復号

Sun-3 では，1回の暗号化に，ハードウェア実装の DES では 0.5 ミリ秒(ms)，ソフトウェア実

装の DES では 1.2ms かかる．したがって，1 回の RPC につきハードウェアでは 2ms，ソフトウェアでは 5ms だけ余分な時間がかかることになる．NFS では，要求を出してから応答が返ってくるまでには平均で 20ms かかるので，ハードウェアの場合は処理の 10%が，ソフトウェアの場合は処理の 25%が暗号処理に使われている．

公開鍵暗号の処理は DES よりもずっと時間がかかる．これが Secure RPC において公開鍵暗号系と対称鍵暗号系が併用されている理由である．

Secure RPC が機能するためには，サーバとクライアントの両方が共通の鍵を算出しなくてはならない．Sun-3 では算出には 1 回につき約 1 秒かかる．サーバがこの操作を行うのはマウント要求を受けてからであるから，結局 SecureNFS マウントを実行するのには約 2 秒余計にかかることになる．しかしこの Sun-3 における 2 秒の遅れ（Sun-4 なら遅延はずっと小さい）をユーザが感じることはあまりない．NFS マウントの多くはバックグラウンドで行われるため，その間ユーザはメールを読んだり，コーヒーを入れたり，その他仕事にかかる前の準備作業をすることができる．

また，Sun-3 における Secure RPC 処理の 10 ないし 25%の性能低下も，ユーザにはそれほどには感じられない．ユーザはローカルな操作，通常の NFS 操作，そして Secure RPC 操作の 3 つを混ぜて使っているので，実際のスループットの低下はその点を加味した加重平均値にとどまる．

より進んだ勉強のために

Denning, Peter J., ed., *Computers Under Attack: Intruders, Worms, and Viruses*. Reading, Mass: Addison Wesley, 1990.

Kent, S. *Privacy enhancement for Internet electronic mail: Part I—Message encipherment and authentication procedures [Draft]*, RFC 1113, Aug. 1989.

———, *Privacy enhancement for Internet electronic mail: Part II—Certificate-based key management [Draft]*, RFC 1114, Aug. 1989.

———, *Privacy enhancement for Internet electronic mail: Part III—Algorithms, modes, and identifiers*, RFC 1115, Aug. 1989.

Security Study Committee (David D. Clark, Chairman), National Research Council, *Computers at Risk: Safe Computing in the Information Age*. Washington, D.C.: National Academy Press, 1991. ISBN 0-309-04388-3, 320 pp.
(書店または(800)624-6242 への注文により入手可能)

Chapter 11

オートマウンタ

11.1 /etc/fstabの問題点

NFSやNISは，遠隔のファイルシステムを見つけ出したりマウントしたりするための，非常に柔軟な方法を提供している．しかし，非常に大きなネットワークでは，すべてのファイルシステムが同時にマウントされているようにする必要はないし，ネットワーク内のすべてのシステムをユーザが意識する必要もない．オートマウンタ(Automounter)はこのような問題の解決を狙ったものであり，また特権をもたないユーザでも必要に応じてファイルシステムを透過的にマウントすることができるという大きな利点ももっている．

オートマウンタの詳細を見る前に，NFSだけを使っている環境を振り返ってみよう．各クライアントシステムはファイルシステムテーブル(/etc/fstab)をもっていて，マウントする必要のあるファイルシステム――自ホスト内のものから，NFSやAT&TのRFSのような他のシステムまで――の位置が書かれていなければならない．/etc/fstabにファイルシステムを明記する代わりにmountコマンドを使うこともできるが，あまり便利ではないし，ルートの権限を必要とする．

他のコンピュータとのファイルシステムを/etc/fstabに明記してしまうことにはいくつかの問題がある．まず，リモートシステムを移動しようとする場合には，ネットワーク管理者は(あるいは望ましくないことだが，ユーザが)すべてのクライアントを変更しなければならない．例えば政策データベース(policies database)といった，企業全体で広く使われる情報サービスのようなものを考えてみよう．このデータベースは，最初は200メガバイトのディスクをもつSun-3にインストールされたかもしれない．登録企業が増えるに従い，データベース内の政策の数は増加していく．そして，ネットワーク管理者は4ギガバイトのディスクをもつSun-4を使いたいと思うようになったとしよう．そうするためには，おそらくすべてのクライアントの/etc/fstabを書き換えなければならないだろうが，それは容易ではない．特に，ネットワークに100以上ものノードがつながっていたらそれは不可能に近い．

第2の問題点は，mountコマンドも/etc/fstabファイルも複製されたファイルシステム(re-

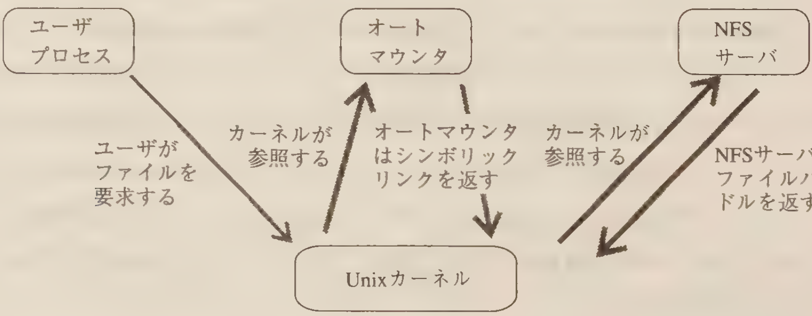
plicated file system)を扱うことができないということである．例えばオンラインマニュアルなどでは，ネットワーク内にファイル階層の実体がいくつも存在する場合がある．どれか1つを主要たるマニュアルサーバにして，他をバックアップとして使えたら便利ではないだろうか．

最後に，要求されたときだけファイルシステムをマウントする必要があげられる．ファイルシステムをずっとマウントしておくことは，資源を消費する．先に見たように，NFSのプロトコルは状態をもたないことを基本にしているので，必要とされる資源は最小限度である．それでも，アクセスする可能性のある数百ものファイルシステムを相手にしては，必要とされる資源は増加する．さらに，100以上ものノードをもつ大きなネットワークでは，時が経てばファイルシステムの位置(location)が変わるのは避け難いことであるが，そのたびにアンマウントと再マウントの操作が必要とされる．

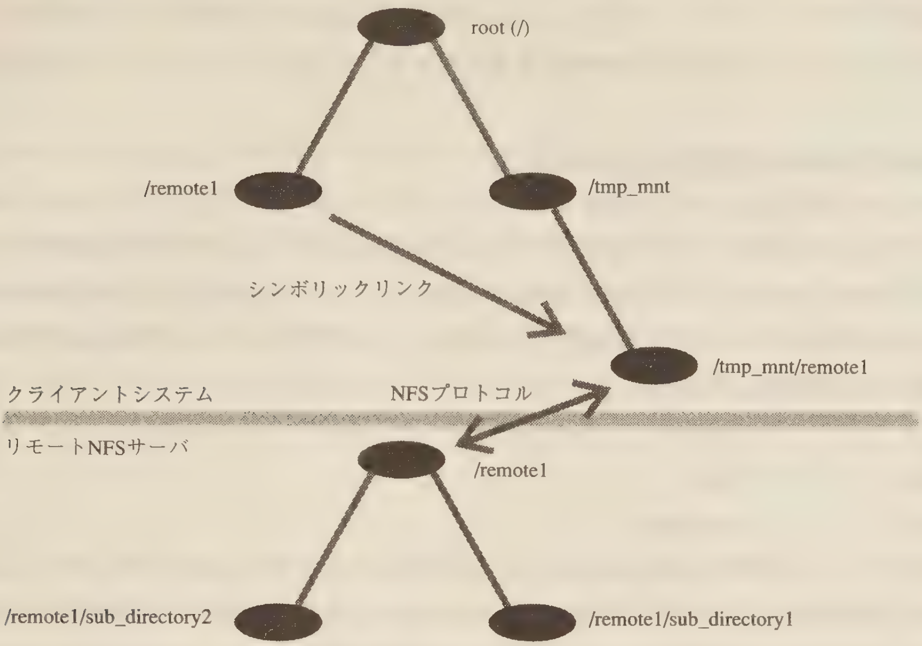
11.2　オートマウンタの概要

オートマウンタは，クライアントのワークステーションでNFSファイルサーバをエミュレートするプログラムであり，固定的なマウントに比べてはるかに柔軟なファイルシステムを提供する．カーネルはリモートホストアドレスの代わりにローカルホストアドレスを使って，リモートファイルサーバに要求を出すのと同じようにオートマウンタのデーモンに要求を出すことができる(図11-1)．

オートマウンタは，ローカルファイルシステムやNFSサーバによってサービスされる他のマウントポイントと同じように，あるマウントポイントをサービスするものとして自分自身をカーネルに登録する．要求がそのようなマウントポイントの1つに入ってくると，それはオートマウンタに渡される．オートマウンタはまずデータベースを参照して，要求されたファイルシステムをどのリモートサーバがもっているかを見い出す．そして，本当のマウント要求をそのリモートシステムに対して出し，カーネルに対しては，たったいまマウントされたファイルシステムへのシンボリックリンクを返し，データはそちらへ取りにいくよう指示するのである．



11-1　オートマウンタの動作



11-2 マウントポイント

図11-2に，オートマウンタがどのようにローカルファイルシステムをだますかが示されている．ユーザは，/remote1と呼ばれるディレクトリにアクセス要求を出す．ところがオートマウンタが自分自身をそのファイルシステムをサービスするNFSサーバであると登録しているので，カーネルはオートマウンタに/remote1に対する要求を出す．オートマウンタは内部のデータベースを参照し，そのファイルシステムはリモートサーバであるserver1が公開しているファイルシステムであることを知る．

そこで，server1:/remote1をマウントする要求が出される．実際のマウントポイントは/tmp_mntと呼ばれる一時的なディレクトリの中である．ユーザはこういった詳細を気にしたくはないだろうから，ユーザの見る/remote1から実際のファイルシステムへシンボリックリンクが張られる．オートマウンタはそのシンボリックリンクを返す．

この時点で，カーネルは多くの場合/tmp_mnt/remote1へとシンボリックリンクをたどる．そのマウントポイントはリモートNFSサーバによって処理され，ユーザの要求(例えば，ディレクトリ中のファイルを一覧する)を満たすために一連のNFSパケットがやりとりされることになる．

いったんシンボリックリンクが返されると，オートマウンタはもはや関わりをもたなくなることに注目してほしい．ユーザがファイルを読み込んだり，サブディレクトリに移動したり，そのほかの通常のファイル操作を行うのにリモートファイルサーバと直接やりとりするのである．

しばらくすると，ユーザはそのファイルシステムへのアクセスをやめる．オートマウンタはアクセスを監視しており，ある時間(デフォルトでは5分)アクセスが途絶えたら，リモートファイル

システムをアンマウントする．それでも，ユーザは再び/remote1に移動することができる．その場合，オートマウンタはserver1:/remote1を再マウントして，再びシンボリックリンクを返す．

11.3　NFSサーバのエミュレート

オートマウンタはNFSプロトコルのサブセットを実装している．ファイルに実際にアクセスするためのcreate, remove, setattr, read, write, renameといったコールはオートマウンタには必要ない．その代わり，オートマウンタはカーネルがファイルやディレクトリを探し出すための，次のようなコールを用意している．

- null
- lookup
- getattr
- readdir
- statfs
- readlink

ユーザがファイルを探すためには上記のコールのどれかを必ず使う必要があるので，これだけのサブセットを用意しておけば，そこに本物のファイルシステムがマウントされているのだとカーネルに思わせるには充分である．これらのコールの戻り値は他のファイルシステムへのシンボリックリンクである．

オートマウンタは，まずUDPのループバックソケットをオープンし，自ホスト内のポートマッパサービスに自分自身をNFSサーバであると登録する．それから，オートマウンタサービスデーモンをforkして，UDPソケット上のNFS要求をlistenする．

その一方で，オートマウンタコマンドは自ホスト内のデータベース(一連のNISマップとローカルファイル)を参照し，一連のmountシステムコールを発行して，特定のマウントポイントを処理することをカーネルに知らせる．mountシステムコールはファイルシステムの種類によって異なるが，オートマウンタは通常次のようなコールを用いる✣．

```
mount("nfs","/usr",&nfs_args)
```

実際のコールでは，nfs_argsはNFSサーバのネットワークアドレスを含んでいる．ところが，そのネットワークアドレスは必ずしもIPアドレスでなくてもよく，ローカルプロセスでもよいわけで，今の場合はそれがオートマウントデーモンである．というわけで，われわれはループバックバージョンのRPCプロトコルを用いてオートマウンタデーモンと接続するわけである．

ここまでのことをまとめると，もしあるディレクトリ(例えば，/usr)を見たければ，ローカルプロセスであるオートマウンタと連絡をとれということである．それは/usrファイルシステムと同じように振る舞うが，重要な相違点は常に実際のデータではなくシンボリックリンクをカーネルに返すということである．

✣(訳注)　例は原文のまま．正しくは，int mount(char *type, char *dir, int flags, caddr_t data)

11.4 マップ

オートマウンタの設定は，マウントすべきファイルシステムの名前とマウントされる場所を格納している一連のマップにより扱われる．オートマウンタが起動されたとき(普通は rc. local の中で)，マップの名前を受け取る．

```
automount /src /etc/srcmap
```

このコマンドは，/src ディレクトリをマウントポイントとし，どのリモートファイルシステムがこのポイントからアクセスされるかを知るのにファイル/etc/srcmap を使え，ということである．

オートマウンタに使われるマップには，次の 3 種類がある．

・ファイル

・NIS マップ

・“組込み”マップ

組込みマップは後で詳しく議論するが，/etc/hosts などの既存のファイルを用いてローカルファイルシステムが拡張されたように見せる機能を提供する．ファイルと NIS マップを用いることによって，ネットワーク全体で共通のオートマウンタの動作(NIS マップによる)とローカルな変更(ファイルによる)とを組み合わせることが可能となっている．

automount のコマンドラインで指定されるマップはマスタマップである．それは別の種類のマップを指していてもよい．例えば，マスタマップは/etc/auto.master というファイルに格納されている．

マスタマップは基本的には他のマップのリストであり，3 つのフィールドから構成される．

```
mount_point   map_name   [mount_options]
```

マウントポイントはディレクトリの完全なパス名である．もしそのディレクトリが存在していないなら，オートマウンタがそれを作成する．ディレクトリが存在しているなら，そのポイントにマウントすることでもとのディレクトリの内容は隠される．

マスタマップで指されるマップは直接マップでも間接マップでもよい．どちらも，次のようなフィールドを有している．

```
key   [mount_options]  location
```

直接マップと間接マップの唯一の違いは key である．直接マップでは key は完全なパス名であるが，間接マップでは key は単なる名前である．

直接マップの標準的なエントリは次のようなものである．

```
/usr/man   -ro,intr   manual:/usr/man
```

このエントリによって，オートマウンタはマウントポイント/usr/man を作成する．ユーザがこのポイントを参照すると，オートマウンタは読込み専用，割込み許可モードで manual:/usr/man を

マウントする.

間接マップのエントリの標準的な例は次のようなものである.

```
parsley   -ro,intr   veggies:/usr/greens
```

このエントリにより，オートマウンタは parsley というサブディレクトリが実際は veggies:/usr/greens をマウントしたものであるということを知る．では，parsley はローカルファイルシステム中のどこに置かれるのであろうか．間接マップはあるマウントポイントからの相対的なマップなのである．コマンドラインか他のマップによってその情報が提供される．例えば，マスタマップが次のような行を含んでいるかもしれない.

```
/veggies   /etc/auto.veggies   -ro,soft,nosuid
```

すると,/veggies がマウントポイントとしてとられ，間接マップである auto.veggies のすべてのエントリがその下に追加される．結局こういうことになる.

```
/veggies/parsley  →  veggies:/usr/greens
```

では，ここで標準的な auto.master ファイルを見てみよう.

```
# Mount-point   Map                 Mount_options
/-              /etc/auto.direct    -ro,intr
/home           /etc/auto.home      -rw,intr. secure
/net            -hosts
```

このマスタマップは 3 つのエントリを含んでいる. 1 つ目の/-というマウントポイントは, 単に欄を埋めるためのものであり，完全なパス名が auto.direct というマップに書かれていることを示している．次の/home というマウントポイントは，auto.home というマップを指している．最後の/net は特別な組込みマップである“hosts”を指している.

11.4.1　間接マップ

代表的な間接マップの例として，auto.home を見てみよう.

```
jones     willow:/home/jones
pynchon   cypress:/home/pynchon
malamud   malamud:/home/malamud
```

なぜこのようにするのだろう．例えば，通例としてすべてのユーザのホームディレクトリを次のような形にしたとしよう✣.

```
/home/node_name/user_name
```

ネットワークを通じて，パスワードデータベースで各ユーザのホームディレクトリがこのような形になるようにしておく. Unixでは, ユーザが ~usernameというディレクトリを参照すると, それ

✣(訳注)　原著のこの例はややわかりにくい. willow というマシンに jones というノード名を付けているということらしい.

はそのユーザのホームディレクトリに置き換えられる．例えば，TomはノードPynchonにホームディレクトリをもっているとしよう．するとローカルな環境で，次のようなコマンドが実行できる．

```
othernode%   cd ~tom
othernode%   pwd
/home/pynchon/tom
```

cd(change directory)コマンドで，現在のワーキングディレクトリを/home/pynchon/tom に移動することができる．auto.home マップを用いると，このようなことがネットワーク環境で実行できるようになる．

```
othernode%   cd ~carl
othernode%   pwd
/home/malamud/carl
```

~carl の参照は，まず(パスワードデータベースを参照して)/home/malamud/carl に置き換えられる．次に，カーネルはそのディレクトリに移動しようとするが，/home はオートマウンタのマウントポイントなので，その操作はオートマウンタデーモンに割り込まれる．デーモンは auto.home マップを参照し，リモートファイルシステムをマウントして，シンボリックリンクを返す．

このような名前のつけ方は，オートマウンタといっしょに用いられると，ユーザに他のユーザのホームディレクトリに自由に移動することを可能にする．

```
othernode%   cd ~joe
othernode%   pwd
/home/joe_node/joe
```

この時点で，2つの但書きが必要である．まず，ネットワーク環境でこのようなことを行わせるためには，パスワードデータベースが NIS により管理されていて，ユーザがネットワークのどのノードからでもホームディレクトリを見つけ出せるようになっていなければならない．NIS がオートマウンタとともに使われないのならば，ネットワーク管理者がオートマウンタのマップの各ノードでのコピーを同一に保つようにしなければならない．これでは，/etc/fstab からたいして進歩していない．NIS とオートマウンタは一緒に使われるべきである．

2つ目の但書きは，オートマウンタは確かにマウントしようとするが，それが成功するとは限らないということである．ユーザ Joe は自分のファイルシステムを公開(export)しておかなければならない．またユーザがそのディレクトリに移動できたとしても，Joe が自分のファイルに全員に対する読出し許可を与えているとは限らない．

11.5 複数マウントと複数配置

直接マップでも間接マップでも，拡張構文によりオートマウンタを手動マウントよりもはるか

に柔軟に用いることができる．次のマップを見てみよう．

```
/usr/local \
  /bin     ivy:/export/local/sun3 \
  /share   ivy:/export/local/share \
  /src     ivy:/export/local/src
```

これは基本的には1つの長いエントリであるが，バックスラッシュを用いて見やすくしてある．この中には3つの異なるマウントポイントがある．/usr/local/bin，/usr/local/share，そして/usr/local/srcである．それぞれのマウントポイントが別々のサーバにあったり，別々のオプションをもっていたりしてもよい．

なぜこのような複数マウントが必要なのであろうか．複数マウントでは，エントリのうちの1つがユーザによって参照されると，3つすべてのポイントがマウントされるのである．もし3つのエントリが別々の行に書かれていたら，それぞれは自分が明示的に参照されるまでマウントされないであろう．3つのエントリが別々の行に書かれていたとしよう．

```
/usr/local/bin     ivy:/export/local/sun3
/usr/local/share   ivy:/export/local/share
/usr/local/src     ivy:/export/local/src
```

ユーザが/usr/local/binに移動すると，その1つのファイルシステムだけがマウントされる．この時点で，ユーザは次のようにタイプするかもしれない．

```
mynode%cd ../src
../src: No such file or directory
```

cdコマンドによって，まず親ディレクトリ（“..”）に移動し，それから子ディレクトリであるsrcに移動したかったのであるが，この時点ではsrcはマウントされていないために移動できなかったのである．複数マウントでは子ディレクトリの1つが参照されれば，ディレクトリ階層すべてがマウントされる．つまり，ディレクトリ階層のそれぞれのポイントにマウントポイントがあるようなものである．

11.5.1 マウントポイントとオプション

それぞれのマウントポイントには，マップ中でマウントオプションを指定することができる．複数マウントの場合は，すべてのマウントポイントにデフォルトとして指定するオプションと，マウントポイントごとにそれに優先するオプションを指定できる．

```
/usr/local -ro,soft \
  /          node1:/export/local \
  /bin       node2:/export/local/sun3 \
  /share     node3:/usr/local/share \
```

```
/src                 -rw,secure   node4:/home/jones/src
```

この場合は，ディレクトリ階層全体に有効なデフォルトのオプションはソフトマウントと読出し専用モードであるが，ソースディレクトリ(/src)だけはSecure NFSオプションつきの読み書き可能モードになる．

11.5.2 多重配置

多重配置により，オートマウンタは特定のファイルシステムのためのサーバの中で最初に利用可能なものを選んでマウントすることができる．次のマップを見てみよう．

```
/usr/local \
   /        node1:/export/local \
   /bin     node2:/export/local/sun3 \
   /share   node3:/usr/local/share \
   /man     node4:/usr/man \
            node5:/usr/man \
            node6:/usr/man
```

/usr/localの下の/,/bin,/shareにはそれぞれnode1からnode3までの特定のファイルシステムがマウントされる．しかし，マニュアルページ(/man)は3つの異なるサーバに置かれており，ユーザはそのうちどれが選ばれるかは気にしない．

多重配置は，ファイルシステムが同一である場合のみ意味がある．読出し専用データベースは簡単な例であるが，システムのバイナリも多重化されるかもしれない．このファイル階層のどこかが参照されると，オートマウンタは3つのサーバそれぞれにnull NFSコマンドを使って探りを入れる．最初に応答してきたサーバがマウントされる．

この手法は，ごく簡単な負荷分散の1つである．というのは，最初に応答してきたサーバが利用可能な資源を最も多くもっていると仮定できるからである．サーバが忙しければ，すばやく応答することはできないだろう．しかし，この方法はやや不充分である．その時点でどのサーバが最も暇かを見ているだけであって，長時間にわたる平均負荷や処理能力を計測しているわけではないからである．

多重配置は次のようにも書ける．

```
/usr/man   -ro,soft   node1,node2,node3:/usr/man
```

この構文は，多重配置がすべて同じパス名で利用可能であることを示している．

ここで，node1が最初に応答してきてマウントされたとしよう．その後で，node1が使用不能になったらどうなるだろうか．デフォルトでは，オートマウンタは5分間ファイルシステムが活動しなかったらアンマウントする．そこで，ユーザが5分間そのファイルシステムを参照しなかったら，それはアンマウントされ，次に/usr/manを参照すると別のノードが選ばれることになる．

5分間待つ代わりに，使えなくなったファイルシステムを明示的にアンマウントしてもよい．再びそこが参照されたとき，ダウンしたサーバはRPCに応答しないので，他のノードが代わりに選ばれる．

11.5.3 サブディレクトリフィールド

マップのもう1つの構文によって，（マップを書く人以外についても）ユーザの時間を節約することができる．マップの正確な構文は実は以下の通りである．

```
key    [mount_option]    server:pathname[:subdirectory]
```

例えば，次のようなのは有効なマップである．

```
john    home_node:/home/home_node:john
mary    home_node:/home/home_node:mary
joe     home_node:/home/home_node:joe
```

ユーザが~johnを参照すると，オートマウンタは2つのことをする．

- home_node:/home/home_nodeをマウントする
- /tmp_mnt/home/home_node/johnを指すシンボリックリンク/home/johnを作る

ユーザが~maryを参照すると，すでにそのパスはマウントされている．そこでオートマウンタは単にシンボリックリンクを加えるだけでよい．余分なマウントを行わないので，ユーザへの応答時間が速くなる．

11.5.4 置換

われわれはこれまで何回か，ホームディレクトリを/homeに置くことにより，変更を容易にしておく慣例を見てきた．しかしこの慣例に従っていると，ユーザの人数が非常に多い場合，マップも非常に大きくなってしまう．これに対して2つの慣用によって，大きなマップのエントリを減らすことができる．まず，エントリ中のアンパサンドはkeyに置き換えられる．

```
john    home_node:/home/home_node:&
mary    home_node:/home/home_node:&
joe     home_node:/home/home_node:&
```

すべてのユーザが自分のコンピュータをもっていたとしたらどうなるだろう．その場合，マップのエントリはすべて次のような形になる．

```
node    node:/home/node
```

このとき，マップはもう1つの慣用であるワイルドカードによって，1行にまでまとめることができる．ワイルドカードはどんなエントリにも一致する．

```
*    &:/home/&
```

いまやユーザは，名前を知ってさえいればネットワーク内のどのホームディレクトリにも移動す

ることができる．ワイルドカードは常に照合に成功するので，マップ中の他のエントリの後に書かれなければならない．

11.6 組込みマップ

ホストマップはオートマウンタに組み込まれた特別のマップである．エントリ内で“-hosts”として参照されるこのマップは，/etc/hosts ファイルと NIS の hosts.byname マップから構成されるホストデータベースを用いている．その内容は，ネットワーク内のホスト名のリストである．代表的な-hosts の利用法は次のようなものである．

```
mynode% automount /net -hosts
```

もちろん，コマンドラインで指定するのではなく，マスタマップや他のマップ中の行で上のように指定してもよい．

ユーザが/net/hostname に移動すると，オートマウンタはそのホストが公開しているすべてのアクセス可能なファイルシステムをマウントする．/net/hostname がマウントされたからといって，hostname に存在するすべてのファイルシステムが見えるわけではない．公開されている部分だけが見えるようになるのである．公開されたファイルシステムは，リモートホストのファイル階層に従って構造化されて見える．もし階層の途中に公開されていないファイルシステムがある場合は，オートマウンタが必要なディレクトリを作成する．

ユーザが/net/hostname に移動したとき，オートマウンタがまず最初にすることは，null コールを送ってそのホストが立ち上がっているかどうかを確かめることである．ホストが立ち上がっていたら，オートマウンタは公開されているファイル階層のリストを送るようサーバに要求する．

次に，オートマウンタはパス名の構成要素の数でそのリストをソートする．

```
/usr/src
/export/home
/usr/src/sccs
/export/root/blah
```

このようにソートしておけば，/usr/src/sccs のマウントより前に/usr/src がマウントされるといったように，正しい順序でマウントが行える．すべてのファイルシステムがマウントされたら，ファイル階層の頂上へのシンボリックリンクが返される．

1 つか 2 つのファイルを求めているユーザにとっては，公開されているすべてのファイルシステムをマウントすることは最適とはいえない．-hosts データベースの目的は，ユーザが自分が何を求めているかわかっていない多くの場合を取り扱うことである．特定のものが繰り返し参照されるようならば，それをマップに含めることを考えるべきである．

-hosts マップとホームディレクトリの手法により，ユーザはネットワークに関して 2 つの知識

をもつことができる.

・ノードのファイルが見たければ, /net/nodename を参照する.

・ユーザのファイルが見たければ, /home/node/username を参照する.

これら 2 つの情報が与えられれば, 誰がどのノードにいるのかを心配することなしにユーザはネットワークの中を渡り歩くことができる.

11.7 オートマウンタを使うべきとき

多くの環境では, オートマウンタのマップはネットワーク管理者によって作られる. その目標は, ユーザがどのワークステーションにいても, 自分のファイルを他のノードから誤りなく手でマウントする必要なしに見ることができるようにするということである. -hosts マップのようなトリックと, 環境変数や置換を注意深く使用することによって, マスタマップのほんの数行でほとんどすべての状況をカバーすることが可能である.

オートマウンタの 2 つの特徴が, "生の"NFS より使いやすくしている. まず, マウントの位置を複数指定できることで, ユーザは重要なファイルシステムの一時的なダウンから隔離される. ファイルシステムの 1 つのコピーが使用不能でも, オートマウンタが自動的に次を参照してくれるからである. また, 要求に応じてファイルシステムを再マウントするので, ファイルシステムの移動も容易になる.

第 2 に, オートマウンタはユーザがすべてを把握することのできないような巨大なネットワーク環境をも扱うことができる. 単にノードの名前を指定するだけで, オートマウンタは利用可能なファイルシステムを選び出し, そのコンピュータのファイルをユーザに利用可能にする. ネットワーク管理者は, 何から何までシステムを組み立てておかなくても,「そのソフトウェアは server5 にあるよ」といった具合に特定のノード名をユーザに伝えるだけですむ.

また, オートマウンタはユーザにとっても非常に有用である. というのは, マップを作るのにスーパーユーザの権限は必要ないので, ユーザはマウントしたいファイルシステムを自分で選ぶことができる✣. mount コマンドによるマウントや, /etc/fstab によるマウントでそうしようとすると, すべてのユーザがルートの権限をもたなければならない.

より進んだ勉強のために

Brent Callaghan and Tom Lyon, "The Automounter," *Proceedings of the Summer 1990 Usenix Conference*, Usenix Association, Berkeley, Calif., pp. 43-51, June 11-15, 1990.

✣(訳注) このようにするためには, スーパーユーザに頼んでマスタマップに自分のマップを参照するようなエントリを作ってもらう必要がある.

Chapter 12

分散アプリケーション

12.1 概要

本章は，外見上は無関係のいろいろな話題について論じる寄せ集めの章の1つである（14章の"Internet アプリケーション"が，もう1つのそのような章である）．これまでのいくつかの章では，1つの種類のアプリケーションを説明するのに1章を費やしてきた．それは，NFS やオートマウンタやメッセージといったものであった．

本章では分散アプリケーション（distributed application）について述べるが，それぞれのアプリケーションに1章を費やすほどのものではない．これは，本章で論じるサービスが他のものに比べて重要でないといっているのではなく，ただ単にそれらについて筆者が述べることがそれほどないといっているだけである．

本章では，まず2つの基盤的なサービスから説明を始める．ライセンスサービス（license service）は，実行可能プログラムをある特定のコンピュータに縛りつけることをしないで，ソフトウェアを使用する許可をベンダが動的に割り当てるための手段である．これは，そこらじゅうのワークステーションにあるのだが，どのユーザにも常時使われているということはめったにないというようなアプリケーションに対しては有用である．

2番目の基盤サービスは，ロックマネージャ（lock manager）である．前に見たように，NFS は状態のないプロトコルである．これは，データのロック機構が NFS のサービスの範囲内では必ずしも与えられないということを意味する．ロックマネージャでは，遠隔ロック機構を加えて NFS の機能を拡張している．ロックマネージャは，複数のアプリケーションがネットワーク全体でロック要求を調停するための手段である．他の Unix 的なロックサービスの場合と同様に，やはりこれも忠告的なサービスである．つまり，ロックを獲得してそれらに従うかどうかはアプリケーション次第である．

ロックマネージャの次は，ユーザがネットワーク内の他のコンピュータにアクセスできるようにする一連のコマンドについて見る．まず，遠隔実行（Remote Execution：REX）という機能に

ついて説明する．これは，適切な権限を与えられたユーザがネットワークから CPU サイクルを借りてくることを可能にする機能である．SunOS に残っているバークレー Unix の遺産の 1 つである r* コマンドについても論じる．ユーザは r* コマンドを使って，ネットワークを介してログインやコピーやその他のサービスを実行することができる．

最後に，PC-NFS について見る．PC は，アーキテクチャや設計が全体的に不充分なので，NFS や Unix ネットワークと統合するための特別な支援機能が必要である．例えば，RPC ではユーザの名前を認証の基本として用いていることを読者は覚えているだろう．それには，ログインというほとんどの PC が提供していない機能の存在が暗に含まれている．ここでは，PC において特別な扱いが必要となる他の特徴もいくつか示すことにする．

PC-NFS に実装されているのは，NFS そのものだけではない．ONC（NFS，RPC，XDR）と DOS オペレーティングシステム用の TCP/IP を実装したものである．PC を統合するときの問題については，16 章で相互運用性（interoperability）について調べるときに改めて見直す．

12.2 ライセンスサービス

ライセンスサービスとは，ユーザにソフトウェアプログラムを利用する権利を与える一連のソフトウェアパッケージのことである．このサービスは，一見不必要に見えるかも知れない．しかし，大きく高価な製品を使用している環境を考えてほしい．例えば，Ingres データベースシステムを利用しているとしよう．10 人の人がシミュレーションを書いており，5 つの大きな Sun のサーバがあるとしよう．ディスク空間や，計算要求や，他の尺度に関してシステムが大きくなると，それが高価になる傾向もあるという結果をもたらす．

わずかの金銭的な検討の結果，ソフトウェアの 2 つのコピーを買う余裕があると考えたとする．そのソフトウェアを 5 つのサーバのどこで動かすのがよいだろうか？　たった 2 つしかないコピーを並行して動かせるように，5 つのサーバのどこででも Ingres を走らせることができれば素晴らしくないだろうか．

ソフトウェア製品をネットワーク全体で柔軟に同時使用するというのは，ライセンスサービスの領域である．同時ライセンスサービスに代わるものとしては，ノードロック機構に基づくコピー保護がある．これは，コンピュータのシリアル番号をソフトウェアに何らかの方法で埋め込んでおき，そのソフトウェアを他のコンピュータ上で動かすことができないようにするというものである．これは柔軟性のない方法であり，ソフトウェアのライセンスが与えられているノードの 1 つまたはそれ以上が長期間オフラインにされている場合，明らかに痛ましいことになる．

ノードロック機構に代わるものとして，サイトライセンスがある．サイトライセンスとは，非常に多額の金と引き替えにベンダが組織の人全員にそのソフトウェアを利用することを許可するというものである．しかし，組織の規模が General Motors と 1 つのコンピュータサイトの中間で

あるなら，サイトライセンスは決して経済的とはいえない．

SunNet ライセンスサービスはこの種の問題を解決している．ライセンスサーバを作動させるためには，ソフトウェアベンダは単純にサイトライセンス機構やノードロック機構で権限の決定をしてしまわないで，ソフトウェアがライセンスサービスを利用するように書く必要がある．

このサービスの基本的な機能は，ライセンスの集合を管理することである．それぞれのライセンスの集合は暗号化されたトークンをもっている．このトークンはソフトウェアベンダによって暗号化されたデータであり，典型的な場合，ライセンス単位（license unit）が含まれている．そして，ライセンス単位は（ライセンス中に埋め込まれている同時に使用可能な限度数を条件として）ネットワーク内のどこででもプログラムを使用できる権利をユーザに与える．

ライセンス単位の概念は，ソフトウェアベンダが自分の望む方法で適用することができる．例えば，ソフトウェアを Sun-4 で走らせるには，Sun-3 より多くのライセンスが必要になるかもしれない．また，新しいバージョンのソフトウェアは，古いバージョンより多くのライセンスが必要かもしれない．

このサービスはネットワーク全体で動作するということに気をつけよう．ソフトウェアは実際にはプログラムの抽象的な集合である．ベンダは容易に，例えば各オペレーティングシステム用のソフトウェアといったようないろいろなソフトウェアを作成することができる．価格表が公表され，特定のシステム用のソフトウェアにはいくつのライセンス単位が必要かが表示される．

ソフトウェアが実行されると，まず最初にそのソフトウェア中にあるライセンスライブラリ (license library) が使用される．ライセンスライブラリはライセンスサーバを見つけて，いくつかのライセンス単位を要求する．ライセンス単位がソフトウェアに与えられると，そのソフトウェアはライセンス単位が正しいものであるかを確かめてから実行を続ける．

SunNet ライセンスサービスにはライセンス要求を認めたり，拒否したり，待ち行列に入れたりする基本機能の他に別の特徴も少し存在する．使用期限の日付がライセンス中に記述でき，限られた期間のデモンストレーションにソフトウェアを利用できるようになっている．ベンダは，例えば CD-ROM といったものでソフトウェアをたくさん配布できる．ユーザは，少しの期間そのソフトウェアを試用することができる．ソフトウェアを試用した後，ユーザはベンダを呼んでライセンスを獲得して，そのソフトウェアをより正式に利用することが自由にできるようになる．

12.2.1 アーキテクチャ

ライセンスサービスには 4 つの構成要素がある．まず初めに，ベンダのアプリケーションとリンクされるライセンスライブラリがある．第 2 にライセンスサーバがあり，これはライセンスデータベースをもつ．3 つ目はライセンス管理ツール（license administration tool）であり，ライセンスを設置したり，その使用状況を監視したりするのに使われる．そして，4 番目にはライセンス生成ツール（license production tool）があり，ソフトウェアベンダはこれを使って実際のライセンス

を生成する．

ライセンスライブラリとライセンスサーバはRPCコールを使って互いに通信し合う．データの受渡しにはXDRを用いているので，ライセンスライブラリがライセンスサーバと異なるアーキテクチャのコンピュータに存在しているかどうかを気にする必要はない．

アプリケーションプログラムがライセンスサーバと通信するために利用できるコールを図12-1に示す．これらのコールは，Network Computing Forumによって開発された現存のAPIと互換性があり，SunNetライセンスサービスと共に動くように書かれたプログラムは，最小限度の変更で他のベンダのライセンスサーバとでも動くようになっていることに注意してほしい．

それぞれのサーバはライセンスの個別のデータベースを管理している．ライセンスは動き回らないということに注意してほしい．つまり，ライセンスはいつも同じサーバに扱われるということである．したがって，サーバが動いていない場合は，そのライセンスは使えない．

これは，アプリケーションの利用が1つのサーバによって規制されるといっているのではない．アプリケーションはネットワーク中を探して，自分が利用可能なライセンスをもっているサーバを見つけることができる．ライセンスが与えられた時点で，アプリケーションは実行を続ける準備ができたことになる．

ネットワークがかなり大きい場合は，複数のライセンスサーバを置いておくことができる．例えば，1つのサーバは営業部門用のものであり，もう1つは技術部門用のものという風にである．ライセンスをどのように（複数のサーバで）分配するかを決定するのは，ネットワーク管理者の責任である．例えば，ユーザが2つのライセンスを利用できる場合，1つずつ別のサーバに置くことができる．またもう1つの選択肢として，両方のライセンスを1つのサーバに配置することもできる．

アプリケーションは探すべき正しいサーバの所在をどのようにして知るのだろうか？　アプリケーション中のライセンスライブラリは，ライセンスを探したときにどのサーバに問い合わせたかを示す結合づけデータベース（binding database）を保持している（これは，ネットワーク中のサーバにランダムに当たっていくのとは対照的である）．結合づけデータベースは，ローカルにもしくはNFSでアクセス可能なファイルであり，製品とサーバの名前を対応づける表が含まれている．サーバの名前はNISもしくはNIS＋のテーブルでも定義されている．NIS＋では，ユーザはそのテーブルに慣習的な情報を付け加えることができる．

1つのホストからライセンスを配るようにしているのは，セキュリティという点からの要件である．ベンダがユーザにライセンスを渡す前に，サーバのホスト名がそのライセンスに埋め込まれる．アプリケーションは，ライセンスを受け取ったときライセンス内のホスト名を見て，それが実際にライセンスを与えたホストの名前と一致しているかどうか調べる．正しいドメインであるかとか，利用可能なライセンス単位の数が正当かなどといった他のチェックもいくつか行われる．

なぜこのようなことを行うのだろうか？　ライセンスを別のいくつかのライセンスサーバに移動して偽装している状況を防ぐためである．この防御は安全なものではない．なぜなら，異なる領域の2つのサーバが同じホスト名とドメイン名をもっている可能性があるからである．もちろんライセンスサーバが正しく作動している一方では，ホスト名やドメイン名の重複という問題を他の非常に多くのアプリケーションがもっているということである．

コール	パラメータ	説明
ls_log_message		ライセンスシステムのログファイルにメッセージを書く．
	my_job_id [IN]	呼出し側プログラムのジョブID
	length [IN]	このメッセージに書かれているテキストのバイト数
	text [IN]	ログファイルに書かれる実際のメッセージ
	status [OUT]	成功，システムエラー，job_idが見つからない，のいずれか
um_get_record		使用状況計測ログ（usage metering log）からレコードを取り出す．サーチ部のいずれかがワイルドカードであってもよい．1回の探索につき1ベンダだけが対象となる．
	my_job_id [IN]	呼出し側プログラムのジョブID（それによって，ユーザと関連するベンダを識別する．）
	job_id [IN]	要求されたジョブのID
	record_id [IN]	要求されたデータベースのレコードのID
	product_id	
	version	
	user	
	group	
	node	
	begin_date	
	end_date	
	maximum_length	返されるデータの最大数
	length [OUT]	返されるバイト数
	data	
	status	
	record_pointer [IN] / [OUT]	複数の値を返すワイルドカード探索で使用される，次に探索されるレコード
um_purge_records		以前にマークされた使用状況計測レコード（usage metering record）を削除する．
	status [OUT]	成功，システムエラー，ジョブIDが見つからない，のいずれか
	record_pointer [IN] / [OUT]	複数の値を返すワイルドカード探索で使用される，次に探索されるレコード

12-1　SunNetライセンスサービスのシステムコール

同時使用コール		
lb_request		特定のアプリケーション用のライセンスを要求する．
	my_job_id [IN]	呼出し側プログラムのジョブID．それによって，ユーザと関連するベンダを識別する．
	product_id	実行される製品の製品ID．
	version	実行される製品のバージョン．
	amount	必要なライセンス単位の数を示す整数．この数のセマンティクスはベンダによる．
	queue	
	check_period	アプリケーションがクラッシュしたとみなす前に，ライセンスシステムがライセンスを利用可能であるが要求されていない状態にしておく秒数．
	license_handle[OUT]	このトランザクションのハンドル．他のコール中に特定のライセンスを参照するのに使われる．
	status [OUT]	ライセンスを獲得した，ライセンスは利用できない（待ち行列に入れられない），利用可能なライセンスがない（待ち行列に入れられる），システムエラー，のいずれか．
lb_wait_remove		ライセンス待ち行列からエントリを取り除く．
	license_handle [IN]	取り除かれる要求のライセンスハンドル．
	status [OUT]	成功，要求が見つからない，システムエラー，のいずれか
lb_release		ライセンスを解放する．
	license_handle [IN]	取り除かれる要求のライセンスハンドル．
	status [OUT]	成功，システムエラー，のいずれか．
出典 ： Recommendations for Network Software Licensing Practices, Version 1.0, The Software Licensing Working Group, Network Computing Forum, June 1, 1989.		
特別なSunNetサーバコール		
nl_init		ライセンスのセッションを開始する．
nl_term		ライセンスのセッションを終了する．
nl_check_que		待ち行列中の位置を更新する．
nl_confirm		ライセンスを更新する．
nl_binding		バインディングファイルを作成させる．
nl_bind_by_vendor		ベンダと製品のライセンスをもつサーバのリスト

12-1（続き）　SunNetライセンスサービスのシステムコール

12.2.2　ライセンス

ライセンスは，ライセンスサーバ内のデータ構造であり，ベンダ名，製品名，バージョン番号，シリアル番号によってそれぞれのライセンスを識別している．利用可能なライセンス単位の数もライセンス中に含まれている．(図 12-2 参照)

一般的な規則では，ライセンス単位は 1 人のユーザを意味しているが，時刻やコンピュータの種類や曜日によって重み付けすることもできる．この方針はソフトウェアベンダによって違う．ライセンスサーバが行うのは，ライセンス単位の管理とそれらのライセンス単位が利用可能かどうかの判断だけである．

ベンダ名
製品名
バージョン番号
ライセンスのシリアル番号
ホスト名
ドメイン名
ホストID
ライセンス単位の数
有効期限の日付（ライセンスが無効になる日）
保守の日付（保守の契約がいつ切れるか）
前のどのバージョンをサポートしているか
ベンダの固有データ

12-2 ライセンスの内容

ライセンス中にはクライアントエントリもあり，そのそれぞれがソフトウェアのユーザ1人を表している．クライアントエントリには，そのユーザに使用されているライセンス単位の数や，ユーザの名前，そのライセンスの次の更新期限が含まれる．

12.3 ライセンスサービスの動作

ライセンスサービスを使用するためにはまず最初に，ソフトウェアベンダがライセンスライブラリを使用する形にアプリケーションを変更することが必要になる．それからそのソフトウェアは顧客に配送される．2番目にやらねばならないことは，ベンダがライセンスを生成することである．これはファイルである．このファイルはソフトウェアのテープに一緒に入れたり，電子メールで送られたり，フロッピーディスクで送られたり，モデムを通じてダウンロードされたりする．ネットワーク管理者はライセンスを取り出して，ライセンス管理ツールを用いてそのライセンスを設置する．それから，ユーザはアプリケーションを走らせようとする．

ユーザがアプリケーションを走らせようとすると，初期化コードがライセンスサーバに対していくつかのコールを行う．1つもしくはそれ以上のサーバにライセンスを要求する．ライセンスが利用できず，行列に入れるのが適切な場合，その要求は1つもしくはそれ以上のサーバで待ち行列に入れられる．待ち行列に入るかどうかの決定はアプリケーション次第であることに注意してほしい．アプリケーションは一方的にこれを決定するかもしれないし，ユーザに尋ねてくるかもしれない．

ライセンスが利用可能になると，アプリケーションは通常通り動く．定期的にライセンスは更新される．最終的にユーザがすべてを実行し終えると，ライセンスは手放される．もちろんアプ

リケーションがうまく動作しなかった場合，ライセンスを手放し忘れるかもしれない．これは，ライセンスを与えられたアプリケーションが走っていたノードが，そのアプリケーションがライセンスを手放す前にクラッシュした場合にも起こる可能性がある．どちらの場合もサーバは次の更新期限が来るまで待って，期限が来たらそのライセンスを他の誰かが利用できるようにする．

ライセンスを手放すのはアプリケーションの責任だが，うまく動作しているアプリケーションがライセンスを手放すのは次の3つの状況のときである．

・アプリケーションが一定期間アイドルになったとき
・アプリケーションがアイコンになったとき
・管理者が戻すように要求したとき

ライセンスを手放すということに関する方針について，興味深い論点がいくつか存在する．例えば，ユーザがそのアプリケーションをクリックした場合，すぐには戻ってこないか，“ライセンスが利用できない”というメッセージとともに戻ってくるかのどちらかだろう．

アプリケーションが一定期間アイドルになったときには，より注意を要する．現在までの仕事の成果を保存せずにユーザが食事に出かけてしまったとしよう．ユーザが戻ってきたとき，ライセンスは利用できない．このユーザは仕事の成果を保存することを許されるべきであろうか？これもまたアプリケーション次第である．これらは，ライセンスサービスをアプリケーションに付加するとき，アプリケーション開発者が考えなければならない（そして，そうするのをできる限り最小限度にしたい）問題である．

12.3.1　ライセンスライブラリの動作

ライセンスライブラリは，ライセンスサーバとの対話を処理し，それらの詳細をアプリケーションに見えないようにする責任がある．ライブラリは，結合づけデータベースに記載されているサーバに問い合わせて，ライセンスを見つける．ライブラリは，利用可能なライセンス単位をもっている最初のサーバから最初のライセンスを受け取る．ライブラリはライセンスを取り出し，それを復号する（ライセンスの一部分は暗号化された形でサーバのデータベースに保存されている）．

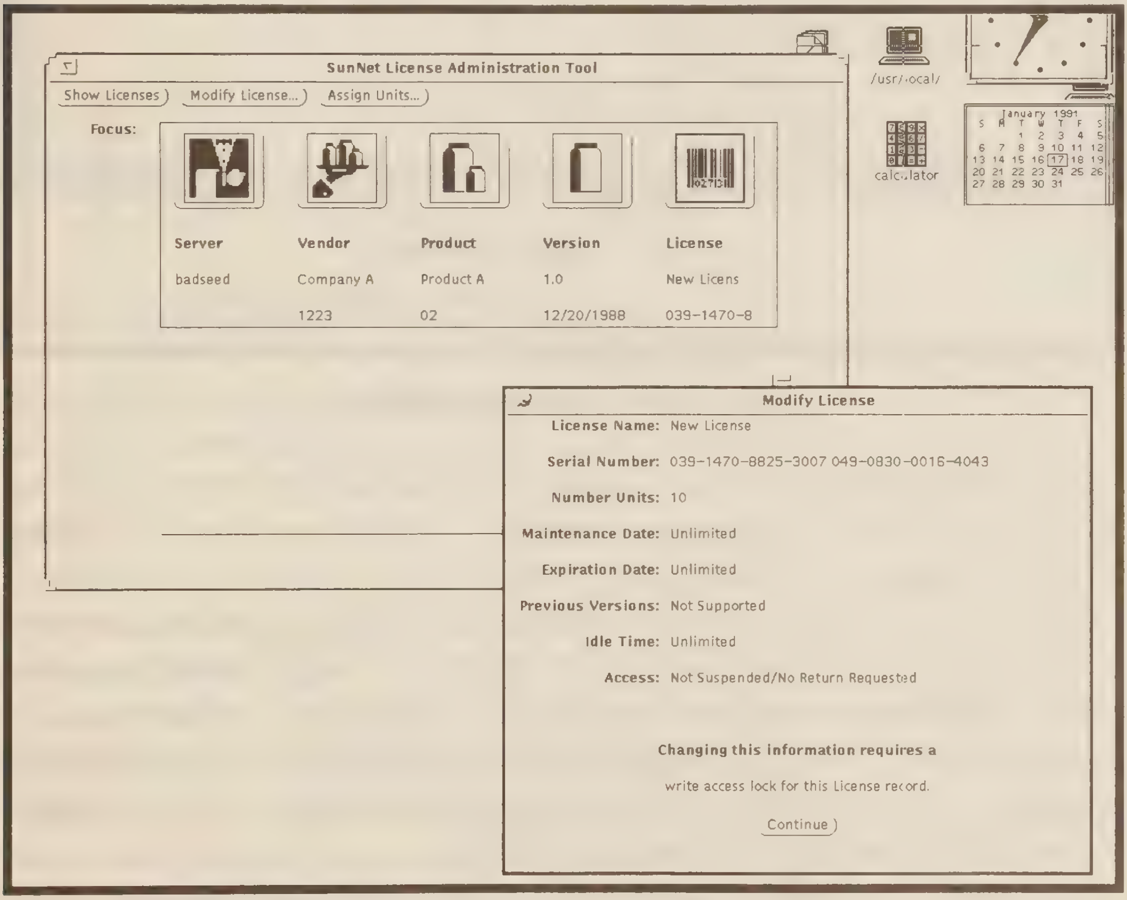
ライセンスの復号操作は，ライセンスアクセスモジュール（License Access Module：LAM）に基づいて行われる．デフォルトの暗号化法もあるが，各自の暗号化法も使用できる．ライセンスが復号されると，そのライセンスの正当性を確かめて，ライセンスが変わっておらずサーバがこのトークンを配布する権利をもっていることを確かめる．ライセンスライブラリは，トークン中の使用中ライセンス単位のいずれかの期限が切れてないかもチェックする．もしどれかの期限が切れていたら，ライセンスライブラリはサーバに連絡し，それらのライセンス単位を解放してプールに戻す．ライセンスライブラリは，ライセンスからベンダの固有データを取り出して，アプリケーションに返すということも行う．

12.3.2 ライセンスの管理

ライセンスサーバのプロセスは，Internet デーモン（Internet daemon：inetd）によって起動される．これは，データベースマネージャとして NetISAM RPC サービスを使用している．NetISAM は，Sun のシステムで利用できる一般的なネットワークファイルサービスである．ライセンスサービス中に NetISAM は束ね込まれている．つまり，NetISAM は一般的な実行時ライセンスではないのである．

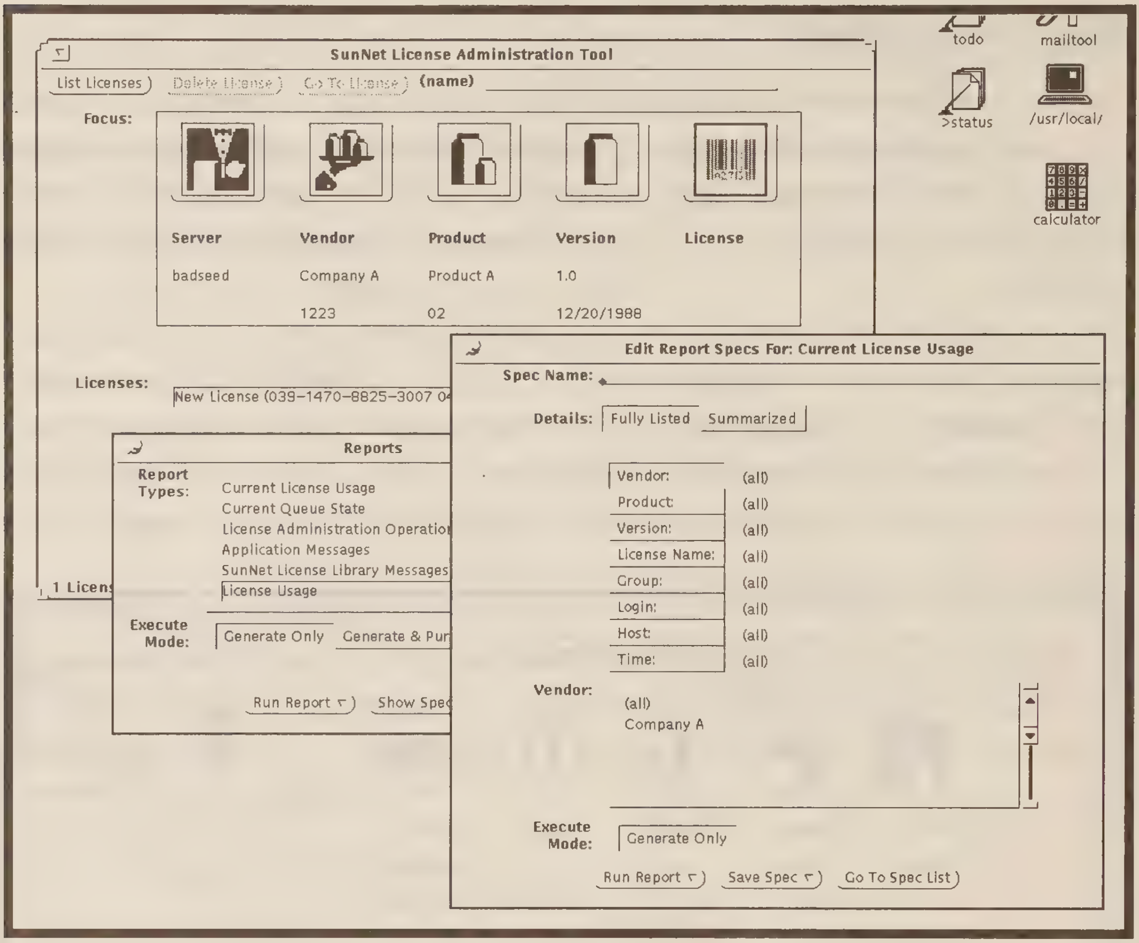
サーバ自体にはライセンスは与えられていない．それは，オペレーティングシステムと一緒にされている．ライセンスサーバの代金は，1 回だけ代金を支払ってライブラリのライセンスを受けるベンダが負担する．その後，ソフトウェアベンダは必要に応じて，サービスを再配布できる．

ライセンス管理ツール（License Administration Tool：LAT）によって，ネットワーク管理者はライセンスを管理したり，設置したりできるようになっている．このツールは，Sun ワークステ



Courtesy of Sun Microsystems

12-3 ライセンスの変更



Courtesy of Sun Microsystems

12-4 ライセンスの使用状況の報告の要求

ーションのビットマップディスプレイで走る SunView や OpenLook などのインターフェイスを用いる．図 12-3 にこのツールの例を示す．この図では，ユーザはサーバ上のライセンスを変更しているところである．

LAT は，ネットワーク上のすべてのサーバを管理するのにも用いることができる．推定できるように大きなネットワークでは，このツールは 2 ヶ所に置かれるであろう．購買部門(もしくはソフトウェアを受け取る人）はライセンスを受け取ったとき，そのライセンスを設置するのにこのツールを使うだろう．ネットワーク管理者もまたこのツールを使ってライセンスの使用状況を監視して，新しいライセンスが必要になったときを決定する(図 12-4 参照)．ネットワーク管理者は，アクセスリストを作成するのに LAT を使うこともできる．

アクセス管理リストは，ユーザやユーザのグループに基づいてライセンス単位を割り当てる．例えばある組織に 21 のライセンス単位がある場合，10 単位を技術部門用に，10 単位を営業部門用



Courtesy of Sun Microsystems

12-5 SunNetライセンス生成ツール

に，残り 1 単位を社長に割り当てるかもしれない．

ライセンス生成ツールは，ソフトウェアベンダの機関で動かすソフトウェアである．このツールは，昔の Unix のリリースに由来する文字処理用の curses ライブラリに基づいて作られており，ほとんどどんな種類の端末でも動くようになっている．ライセンス生成ツールはアプリケーションのライセンスアクセスモジュールを使って，トークンを暗号化し，ファイルに入れる (図 12-5 参照)．

その後，そのファイルはユーザサイトに移動され，誰かに特定のマシン上に設置される．ベンダがライセンスツールに与えなければならない情報はライセンスサーバの名前と，ドメイン名，ホスト ID であろう．

SunNet ライセンスサービスの例は，Sun 自身の CG3270 というソフトウェアである．このソフトウェアは，Sun ワークステーション上で IBM のメインフレーム用のカラーグラフィックス端末 3270 エミュレータウィンドウを開くことができるようにしている．このソフトウェアが動き始めると，ライセンスサーバからライセンスを受け取って，(アクセス管理とライセンス単位が利用できることを条件として) すべてのユーザがこのソフトウェアにアクセスすることが可能になる．

12.4 ネットワークロックマネージャのプロトコル

ロック機構は，あるオブジェクトに対して他のプログラムが操作を行うことを防ぐのを可能にするものである．そのオブジェクトとは通常，ファイルやそのファイルの中のレコードであるが，その概念は他の種類のオブジェクトも含むように拡張できる．ロックは，許される同時実行性のレベルを規制する．

代表的なロックは，書込みロック (write lock) である．これは，排他的ロック (exclusive lock) としても知られている．書込みロックを獲得するということは，そのユーザが書込みを終えるまで他の誰もそのオブジェクトにアクセスできないということを意味する．他のロックは読出しロック (read lock) であり，共有ロック (shared lock) としても知られている．データの書込みを行っているときは，データは一貫性がとれていないかもしれないので，他の人がその情報を読み出すことができるのはよくない．読出しロックを用いると，データを読み出すときには複数の人で共有することが許されるが，書込みロックは拒否される．

ロック機構を使えばマルチステートメントトランザクション (multistatement transaction) を行えるようになることに注目しよう．通常，原子的な (atomic) 更新は，データのほんの一部だけを変更するときにしか使えないし，各更新操作は他の更新操作と完全に独立に行われるからである．

原子的な更新を特徴としてもつような環境では，ロックマネージャは実際には必要ない．この環境では，すべての操作はべき等 (idempotent) であると考えられている．つまり，更新の順番が違ったり，更新が繰り返されたりしても，同じ影響を及ぼすということである．

ここで，銀行業務のアプリケーションについて考えてみよう．前のアプリケーションが 1000 ドルの預金を表示していたとしよう．明らかに，現在の預金残高は前回の預金残高に預金額を加えたものになるはずである．プログラムは，前回の預金残高 9000 ドルを（読出し操作で）得て，それに預金額を加えて 10 000 ドルを示すように現在の預金残高を書き換える．

読出しと書込みの間に，他の誰かが同じ口座に預金をしたらどうなるだろうか？　簡単にするために，もう 1 人の預金額も 1000 ドルとしよう．そして，いま 2 つのプログラムがそれぞれ 9000 ドルの預金残高を読み出したところであるとしよう．

それぞれのアプリケーションは 9000 ドルに 1000 ドルを加え，預金残高を 10 000 ドルに変更する要求を出す．ファイルシステムが 2 つの書込み要求をそれぞれ受け取って実行し，最終的な預金残高は 10 000 ドルになってしまう．これは，客が満足する状態ではない．

この例は，2 つの関連する操作で最後のトランザクションの結果が残ってしまったことを示している．また，他方の操作を行わずに一方の操作を行うと，不正確な結果を生み出す．先ほどの例では，データに対して排他的ロックを用いて，変更されようとしているデータを 2 番目のプログラムが読出しできないようにするのが望ましかった．

Unix オペレーティングシステムにおいて，データの完全性（integrity）を守るのは通常，アプリケーションの責任である．これは，ファイルシステムはデータに対するいかなる読出し要求や書込み要求も認めてしまうということである．一貫性を損なう更新操作が起こらないことを保証するために，他のユーザ（そして，おそらく他のアプリケーション）との間で要求の調停を行うことは，アプリケーションの責任である．

他の多くのオペレーティングシステムでは，ロック機構がもっと下のレベルに組み込まれている．例えば DEC の VMS では，ファイルシステムがロック要求の調停を行う．ユーザがファイルをオープンするときには，希望するロックの状態と共にその要求をレコード管理サービス（Record Management Services：RMS）に送る．RMS は，要求が現在行われている他の要求と一貫性がとれていることを確認してから，そのロック要求を認める．

どちらの場合でも，ロックマネージャは独立したプログラムである．ロックマネージャはファイルを管理するものではない．ある抽象的なオブジェクト（それがまた別の種類のオブジェクトをもっているかもしれない）へ同時に起こった要求を管理するものである．ファイルの例では，オブジェクトとは単純にはファイルであり，ファイル中のレコードやブロックもさらに別のオブジェクトである．データベース管理システムにおいてロックの対象となるのは，表，列，欄，データのインスタンス（instance）である．

ここで重要な点は，いずれかのプログラムがロックを強制せねばならないということである．各プログラムは，あるオブジェクトが現在アクセス可能かどうか知るためにロックマネージャと対話する(ロックマネージャは，他のプログラムとも対話している)．ロックが認められれば，そのロックが守られていることを誰かが見ていなければならない．

VMS の例では，RMS が中心の監視役を果たし，すべてのユーザが適切なロックを獲得してからでないと操作が行えないようにしている．Unix の例では，プログラムはロックを獲得してそのロックに従う．つまり，書込みロックを獲得していないレコードに対してデータを書き込もうとしないようにするのである．

12.4.1 ONC のロックマネージャ

ロックマネージャは，分散環境で利用可能な RPC サービスの 1 つである(NFS や REX や他にもいくつか利用可能である)．このマネージャは，ファイルの忠告的なロック機構を提供する．つまり，ロックマネージャは，ロックマネージャを利用することにしているプログラムの間の調停を行う．もちろんプログラムがロックマネージャを使用しない場合，調停はアプリケーションの責任のままである．

このサービスでは System V インターフェイス定義（System V Interface Definition：SVID）の一部であるロックコールを取り上げ，ネットワークを介して利用できるようにしている．NFS を利用して，プログラムはローカルファイルの場合と同じ方法で遠隔ファイルにアクセスできる．

ロックマネージャを用いることで，プログラムはローカルロックサービスの場合と同じ方法（ロックコール）で遠隔ロックサービスにアクセスできるようなる．

SVIDのロックでは，ファイルとレコードのロック機構を定義している．これに加えて，ONCのロックマネージャは，MS-DOSのファイル共有機能をサポートして，PCのアプリケーションにもロックマネージャが利用できるように拡張された．

このネットワークサービスにおいて，現在の状態を保持するのはクライアントにまかされている．クライアントがロックを獲得すると，そのことを覚えておかなくてはならないのはクライアントである．サーバがクラッシュしてリブートされると，クライアントが自分がもっているロックをサーバに要求しなければならない．

クライアントまたはサーバがクラッシュしたことを見つけ出すのは，ネットワークベースのロックや他の状態を保持するプロトコルにとってはきわめて重要なことである．NFSの場合は，プロトコルは状態を保持しないものであることを思い出してほしい．つまり，NFSにおいては，クライアントまたはサーバのどちらかがクラッシュしても回復動作は必要ないということである．

状態を保持するアプリケーションをサポートするために，状態モニタ（status monitor）と呼ばれる別のサービスが作り出された．これは，遠隔コンピュータがまだ動作しているかどうかを判断する責任を負っている．ロックマネージャはロック要求の調停を行っており，状態モニタは遠隔クライアント✣1の現在の状態の情報を集めている．

基本操作はきわめて簡単である．クライアントは遠隔サーバに要求を出す(図12-6参照)．その要求が認められるとクライアントは，一貫性を損なう操作が同時に起こるかもしれないということを気にせずに遠隔のオブジェクトに適切な操作を実行できるものと考える．通常，その後アプリケーションによってロックが解除され，その資源を他のユーザが利用できるようになる．ロックの解除はすばやく行われることが望ましいが，アプリケーションは，そのユーザが長い食事に出かけてしまっている間はファイルをロックしているとみなす．

ノードに問題が起こると状態モニタがクライアントかサーバに知らせる．サーバがクラッシュしている場合，クライアントは待つことを選ぶ．データベースの例では，クライアントはマルチステートメントトランザクション（multistatement transaction）✣2の途中であるかもしれないし，その処理を破棄するか完了するためにサーバに連絡をとることが必要である．

サーバがリブートされたとき，そのサーバはすべてのロックが解除されたきれいな状態で動き始める．サーバは，猶予期間（a grace period）の新しいロック要求はまったく認めない．ただしその期間中に，回復したサーバ上のロックをもっているクライアントは，サーバがリブートした

✣1　このクライアントは，状態モニタに対するクライアントであり，遠隔マシンの状態モニタである．状態モニタどうしが情報を交換し合って，遠隔マシンの状態を知る．状態モニタは特定のプロセスの状態までは監視しない．

✣2　マルチステートメントトランザクション（multistatement transaction）については，巻末の用語集を参照せよ．

Do Nothing	
Test Lock	要求されたロックの記述を受け取り，そのロックがブロックされていないかどうかを返す．
Establish Lock	要求が待ち行列に入れられるべきであったり，そのロックが排他的ロックであったり，そのロックが再要求である場合，ロックするオブジェクトの情報を受け取る．
Cancel Lock	
Unlock File	
Grant a Lock	要求を認めるときにサーバが送る．
Share a Lock	ファイルのアクセスモードを読出し専用，書込み専用，読み書き両用のいずれかにセットするためのDOSの拡張．これらの要求は待ち行列に入れられない．つまり，もし拒否されるとクライアントが後で再度要求を出さねばならない．
Unshare a Lock	
Unmonitored Lock	監視のないロック（unmonitored lock）を設定するためのDOSの拡張．PCにサーバはサーバがいつクラッシュしてリブートされたかを通知しようとはしない．
Free ALL	リブートされたときに，使用中のロックをキャンセルするためにPCはこのコールを送る．

12-6 ロックマネージャ間で使用されるコール

ことを知らされ，以前のすべてのロックを再要求することができる．猶予期間が終わるとサーバは通常の操作を再開し，新しいロック要求の調停を続行する．

サーバがクラッシュした場合には再要求の時間があるのに対して，クライアントがクラッシュした場合には，クライアントが保持しているすべてのロックを解除する．ロックされている資源はサーバ側にあるので，クライアントが回復している間の不確定な期間その資源を使用できなくしておくのを避けるのは有用である．なぜなら，アプリケーションはそれが動いていたクライアントと同時にクラッシュして，結局実行し直さなければならないからである．

基本的なコールは，ファイルやそのファイル中のレコードに対するロックを獲得するためのものである．遠隔手続き呼出し（remote procedure call）中のフラグによって，クライアントは以前のロックを要求できるようになっている．明らかに，このフラグはサーバの猶予期間中のみ意味があるもので，通常の操作中には無視される．

操作が終わると，ユーザアプリケーションはロック解除の要求を出す．ロックが認められるかどうかを調べることも可能である．もちろん，このテストに対するメッセージが戻ってきてロックを獲得しようと決定するまでの間に，他の誰かがそのプログラムを追い抜いてロックを獲得してしまうかもしれない．

ロック要求の身元の確認はNFSの場合と同様に処理される．クライアントは，クッキー（cookie）またはファイルハンドルとして知られている隠蔽された（opaque）データを受け取る．クッキーは，サーバにとって意味のあるものである．例えばUnixサーバでは，クッキーには特定のファイルに対応するiノードのような情報が含まれている．

ロック要求には，長さとオフセットも含まれている．Unixではファイルシステムによってレコ

ードは与えられていないことに注意してほしい．レコードを固定長にするか可変長にするか（可変長のレコードとは，例えば後ろに復帰改行（carrige return）が続くテキスト）は，アプリケーション次第である．ファイルシステムは（長さとオフセットで定義される）データのブロックを与えるだけなので，ロックサービスが同じような（長さとオフセットによる）サービスを提供するのは有用である．

ロックが認められると，2つのデータによって身元が認識される．ファイルハンドルがオブジェクトの身元を表し，所有者ハンドルがデータの所有者の身元を表している．ファイルハンドルがクライアントにとって隠蔽されているものであるのに対して，所有者ハンドルはサーバにとって隠蔽されているものである．

基本的なSVIDコールはUnix環境でうまく動作するが，他のオペレーティングシステムには違ったロック機構のセマンティクスがある．特に，MS-DOSにはファイル共有の概念（これは，脳死してしまっているこのオペレーティングシステムにとっては急進的な新しい概念である）が加えられ，ファイルロック機構と組み合わせて用いられている．ファイル共有は他のユーザができることを指しており，それに対してファイルロックはユーザ自身ができることを指している．したがって，ファイル共有機構のモードは将来同時に起こるロック要求に影響を及ぼす．

この2つを組み合わせた典型的な例は，読み書き両用にファイルをオープン（ロック）しているが，他のユーザがそのデータを読み出すことが許されている場合である．つまり，他人に書込みを行ってもらいたくはないが，データを（それが一貫性がとれてないものであっても）読み出すことは気にしないときである．

12.5 REX

REXはRPC機構の上に作られた遠隔実行のサービスである．これは，コマンドを遠隔のコンピュータで動かすのに使う4.3 BSD Unixのrshコマンドにいくぶん似ている．ただし，REXにはrshに優る重要な点が少し存在する．

REXの背後にある基本的な考え方は，コマンド名が他のコンピュータに送られて実行されるということである．実行結果は，クライアントシステムに返ってきて表示される．REXをいっそううまく使用すれば，ユーザは複数のマシンアーキテクチャをまたいで仕事することが可能になる．

rshとREXを理解するためには，Unixシステムでプログラムがどのように実行されるかを簡潔に見ておく必要がある．Unixにおけるユーザインタフェイスはシェルである．シェルとは，コマンド処理環境のことである．シェルは，一連の環境変数をもっている．環境変数とは，現ワーキングディレクトリや，時刻の表示形式に対するユーザの好みといったものである．

シェルは起動時に，始動手続き記述用のローカルな始動ファイルの中を見る．その内容は，（大部分は）環境変数をセットするコマンド群である．同時に複数のシェルを起動していることは，

まれなことではない．そして，各シェルはいくつかのバックグラウンドジョブやストップしているジョブと現在のフォアグラウンドジョブをもっている．ウィンドウ環境の場合，各々のウィンドウが別々のシェルである可能性がある．

rsh コマンドの考え方は，シェルを遠隔コンピュータ上で起動できるようにするというものである．ユーザは遠隔コンピュータの上のアカウントをもっているか，コマンドラインでユーザ名とパスワードの組を明示せねばならない．

ユーザは，それらのアクセス制御情報に加えて，遠隔ホストの名前と走らせたいコマンドも示さねばならない．rsh コマンドはこの情報を使って，ネットワーク経由の TCP コネクションを張る．事実上，ユーザは遠隔のマシンへログインすることになる．

ただし，これは実際にログインするよりずっと便利である．実際にログインするには，ユーザが数ステップを行う必要があるからである．rsh コマンドでは希望のコマンドを起動して，クライアント（ローカル）コンピュータへ結果を返すまでの間，遠隔コンピュータにログインしていることになる．

しかしながら，ここで重要なことはシェルが遠隔マシンで起動されるということである．これには少し時間を要する．また同様に重要なことは，ローカルな環境変数やローカルファイルは遠隔マシンにわざわざ移動しなければ利用できないということである．

REX はローカルなシェルの拡張とみなすことができる．REX は遠隔のシェルを起動する代わりに，ローカルな環境変数を読み出して，RPC コールを使って遠隔のマシンにそれらを送る．この時点で，(REX デーモンはすでに起動されていなければならないが，) 他のどんなサーバも新たに起動する必要はない．そしてシェルを起動する必要もない．

REX で行うのは，“環境変数を送る”ということだけではない．REX は，NFS を使って遠隔マシンにユーザの現ワーキングディレクトリをマウントする．これで，ローカルなファイルが利用できるようになる．REX は，rsh のように明白に別のマシンにログインするのではなく，現在のマシン能力を補強するために遠隔の CPU を数分の間借りるという感じである．

ワーキングディレクトリが遠隔マシンにマウントされると，TCP ベースのコネクションが張られて，Unix で使われる 3 つの基本チャネルに合致される．その 3 つとは標準入力，標準出力，標準エラー出力である．ワーキングディレクトリをマウントして，基本チャネルを結合することは，REX ベースのコマンドをローカルな命令と組み合わせて使えるので，重要である．例えば，ローカルな操作の結果を REX によって実行されている遠隔操作にパイプで送ることができる．また，エラー出力は特定のファイルにリダイレクトすることもできる．さらに，出力を別の命令にパイプで送ることもできる．

REX が RPC に基づいたサービスであるという事実は重要である（図 12-7 にこのサービスの基礎をなす RPC コールを示す）．REX では，基本 RPC 信任状（RPC credential）がユーザを認証するのに使われている．それに対して，遠隔シェルコマンド (rsh) では，UID に基づいた非常

REXPROC_START	REXのセションを開始する.
REXPROC_WAIT	REXのセションを終了する——クライアントは現セションでこれ以上要求を出さないことを示す．実行を完了するのが可能になり，コマンドの結果の状態が返される.
以下の3つのコールは対話モードで用いられる.	
REXPROC_MODES	端末のモードを初期設定する．（例えば，使用する消去文字や割込み文字を指定する．）
REXPROC_WINCH	クライアント側のウィンドウの大きさの変化を知らせる.
REXPRC_SIGNAL	サーバにシグナルを送る.

12-7　REXで使用されるRPCコール

に簡単な認証法を使っており，本質的にはクライアントホストの名前だけしか見ていない.

REX は RPC の下層で TCP トランスポートサービスを用いている. TCP を使用することで, 大量の引数データをクライアントとサーバの間でやりとりできることになる．それに対して UDP では，データのサイズが 8kbytes に制限される．なぜ REX にはこのすべて(大量の引数データを送れること)が必要なのだろうか？　REX はシェルの環境をすべてサーバに送ることを思い出してみよう.

REX は 5 つの RPC 手続きで構成されている. on プログラム中で各々の手続きが呼ばれ，サーバデーモン（REXd）によって実行される.

最も基本的なコールは，REXPROC_START である．これは，シェルの環境変数と現ワーキングディレクトリと使用するコマンド名などを遠隔コンピュータに送る．これには標準入力，標準出力，標準エラー出力のために使用されるクライアント(ローカル)コンピュータ上の 3 つのポート番号も含まれる．サーバは，コマンドが実行される前にその 3 つのポートにコネクションを張る．そのシステムで REX を動かすためには，インターネットデーモン設定ファイル(inetd.conf)中にあらかじめ記述しておかなければならない.

REX ユーザは，REX サーバに知られていなければならない．REX サーバは，どの権限の下でコマンドを実行するかを示す身元を確認するのに RPC Unix 認証を用いている．これは，両側(サーバとクライアント）が同じ NIS ドメインを共有している（または，両方のマシン上にそのユーザのローカルなパスワードエントリが存在している）ことを仮定している.

Secure RPC バージョンの REX コマンドも存在する. Secure RPC バージョンの REX では，ユーザが権限をもっていると遠隔サーバに認証されるかどうかの責任をもつ．さらに，ファイルが安全にマウントされることを保証するために Secure NFS を使うべきである.

REX には，2 つの要素が存在する．それは，クライアントとサーバである．クライアントマシン上には，“on”コマンドがある．これは，バッチスタイルのコマンドとしても対話型のコマンドとしても使うことができる.

バッチスタイルの場合，コマンドが実行されて結果が戻ってくる．例えばファイル名のリストを見るときは次のようにする.

myhost% on bighost ls

このコマンドラインは，ホスト bighost で ls を走らせるということを指示している．ここで面白いことは，ls コマンドを動かしたときには on コマンドが起動された（ローカルな）現ワーキングディレクトリの内容を表示するということである．ls コマンドにパラメータを与えない場合，現ワーキングディレクトリを指定したことになるからである．（ローカルマシンの）現ワーキングディレクトリは，NFS を使って bighost 上に遠隔マウントされる．

on コマンドの対話型バージョンは，ユーザとの対話を許すプログラムを遠隔ホストで走らせるのに使われる．例えばあるファイルを見るのに，遠隔ホスト上で vi を使うことができる．

myhost% on bighost vi /usr/src/bigfile

このコマンドは，ユーザが vi を終了するまでホスト bighost で vi プロセスを動かす．ここでファイル(/usr/src/bigfile)は，遠隔のホストのものである．これはローカルファイルシステムの/usr/src であるというわけではない．

一見この最後の記述は on コマンドについての最初の記述に反するように感じるかもしれない．on コマンドが遠隔ホストで動かされるときは，ローカルホストの現ワーキングディレクトリだけが NFS を使ってマウントされる．そして，他のすべてのファイルの参照は遠隔のファイルシステムに対して行われる．このように取り決めた理由は，REX がマシン独立性を保つためである．例えば/bin ディレクトリを考えてみよう．このディレクトリには，一般的な Unix コマンドの実行可能ファイルがある．しかしながら各々の実行可能ファイルは，対応するマシンアーキテクチャの命令が含まれている．例えば SPARC ステーションが SPARC アーキテクチャを使っているのに対して，Sun-3 は Motorola の 68000 シリーズのチップを使っている．

それゆえに，デフォルトでは遠隔マシン上のファイルを使うべきである．それは(遠隔ファイルシステムの)いかなるプログラムも遠隔 CPU のアーキテクチャと対応しているからである．ただし，例外が 1 つ存在する．一般に，現ワーキングディレクトリにはデータが存在しているということである．例えば，コンパイル前のソースコードや編集中の文書や既読のメールなどである．したがって，遠隔のファイルシステムと（ローカルファイルシステムの）現ワーキングディレクトリを使うのが賢明である．

いうまでもなく，これらはすべて NFS を創作的にマウントすることで変更することができる．慎重にマウントとクロスマウントを行うことで，必要なファイルを利用することができるようになる．また，オートマウンタをローカルなネーミングの取決めとともに用いることで，必要なファイルが参照時に利用できるようになる．

もう一度 REX と rsh について見てみよう．REX が rsh より実行時間が短い理由の 1 つは，rsh では遠隔ホストで環境変数をセットしなければならないのに対して，REX では環境変数を転送するからであるということを覚えておこう．

ここで，2 つの点に注意するべきである．まず第 1 に，走行時間がより長いコマンドに対して用

いると，この2つのアプリケーション(on と rsh)間の性能の差が減少するということである．これは，セットアップ時間に差があるだけであって，実行速度の違いが存在するわけではないからである．第2に，(REX では)ネットワークを介してデータを転送することは，初期化するより速いと考えていることである．複雑なローカル環境と簡単な遠隔の環境が存在する場合には，転送時間がセットアップ時間を上回るかもしれない．

この2つのコマンドの相違を説明する方法は，環境表示(printenv)コマンドを使ってみることである．このコマンドは環境変数の値を表示するものである．これらのコマンド(on や rsh)は，コマンドの実行時間を計測する/bin/time コマンドと組み合わせて使用することができる．

```
myhost% /bin/time rsh compuhost printenv HOST
compuhost
9.2 real 0.1 user 0.2 sys
myhost% /bin/time on compuhost printenv HOST
myhost
2.6 real 0.2 user 0.3 sys
```

最初のコマンドラインではシステムが rsh コマンドの時間を計測し，rsh コマンドが printenv HOST コマンドを起動するように指定した．したがって2種類の出力が見られる．最初の出力では，変数 host が遠隔のコンピュータ上の“compuhost”という値をもっていることを示している．2番目の出力では，実行に要した時間は実時間で9.2秒であり，そのうちの CPU 使用時間は0.1秒がユーザモードで使用され，0.2秒がシステムモードで使用されたということを示している．

2番目のコマンドラインではシステムが on コマンドの時間を計測し，on コマンドが printenv コマンドを起動するように指定した．ローカルな環境変数が REX によって転送されるので，この結果(printenv HOST の結果)は myhost になっている．実時間の量は(rsh より)著しく少ないが，費やされる CPU 時間は多いということに注意しなければいけない．

12.5.1　REX の管理

REX をうまく動かすために覚えておくべきことが少しある．まず最初に，REX は現ワーキングディレクトリをマウントしようと試みるということである．これは，ワーキングディレクトリ(あるいはそのファイルシステム中のもっと上位のディレクトリ)は，前もって公開されていなければならないことを意味する．

REX が走行するときには，現ワーキングディレクトリ以外のすべてのパス名とファイル名がサーバマシンのファイルシステムのものと解釈される．したがって少し注意しなければならない．例えば，C コンパイラがローカルディレクトリ（現ワーキングディレクトリ）にあるならば，次のコマンドを起動するとそのコンパイラが起動されてしまうだろう．

```
myhost%  on compuhost cc file.source
```

マシン compuhost が異なる種類の CPU をもっているならば，このコマンドは面白い結果を与えることになる．

REX の典型的な例はソフトウェア開発で使われる．VAX と Sun-4 と Sun-3 と SPARC ステーションがあるとしよう．そして，最終的にすべてのマシンに配置されるプログラムを開発しようとしているとする．

複数の種類のマシン上で動かすという異種アーキテクチャに関する問題について悩み始める前に，自分の SPARC ステーション上で(ソース)コードを開発して走らせてみるということは意味がある．(ソース)コードが SPARC ステーション上で正しく動くものになれば，REX を使ってそのコードをそれぞれの遠隔マシンでコンパイルして，各アーキテクチャに対応した実行可能ファイルを生成することができるからである．

REX の別の一般的な用途は，ネットワークの上の動いていない資源を利用することである．例えば，あなたの社長が自分には最新の Sun-4 ワークステーションが必要であると信じて，Sun 2 をシステム管理者であるあなたに渡したとしよう．ほとんどの経営者と同様に，この社長は電話やミーティングに時間のほとんどを費やしている．解決案はかなり容易なものである．ワークステーションをとり変えて，社長が気づかないことを願うというようなことをしなくても，REX によって遠隔資源を使うことができる．大量のデータを整列する必要はありませんか？　もしあるなら，on コマンドを使いなさい(そして，仕事をし続けることができるように，そのコマンドをバックグラウンドで動かしなさい)．

社長が戻ってくると，彼のマシンが少し遅くなっていると気づくかもしれない．これは，天気と気候の状態が社長の CPU の速度にどのように影響を及ぼしているかについてのあいまいな意見によって容易に弁明できる．

12.6 r* コマンド

バークレー版の Unix には遠隔シェル(rsh)に加えて，遠隔ログイン(rlogin)や遠隔コピー(rcp)のような他のコマンドがいくつか存在する．これら 3 つのコマンドすべてが，ローカルなネットワークの境界内でサービスを実行したりデータにアクセスする便利な手段であると考えられている．NFS のような分散ファイルシステムを追加導入して，特にオートマウンタや NIS といった支援サービスと一緒に用いることで，これらの昔のバークレー Unix のコマンドの必要性が大幅に減少するということを認識しておくのは重要である．

ときおり r* コマンドはもっと広い環境で使われる．しかし，セキュリティという点からすると，これらのコマンドは重要なデータがあるネットワークに対して外部からアクセスしようとする疑わしいものとなってしまう．

多くのコマンドにおいて，ユーザがホスト上のサービスにアクセスするのにユーザ名とパスワードを用いる．これは FTP や Telnet のような Internet アプリケーションにあてはまる．11 章で示したように，パスワードを送ることは常に充分であるというわけではないのだが，それは確かに 1 つの防御線である．この防御線は，ローカルネットワーク内では少しめんどくさいものである．ユーザが半ダースものコンピュータのアカウントをもっている場合，それらすべてのコンピュータを 1 つの実行環境として使えるようになることは有用である．

バークレーの r* コマンドでは，このような場合に容易なアクセスができるようにしている．ローカルなサービスにアクセスするのにパスワードを必要としない場合，遠隔ホスト（またはグループ）の名前を/etc/hosts.equiv ファイルに記述しておくことで，そのホストにアクセス権を与えることができる．このファイルは，信頼するホストとしてマシンを認めるのに使用される．例えば，このファイルには次のような行が書かれているだろう．

```
host1
host2 user_a
+@group1
-@group2
```

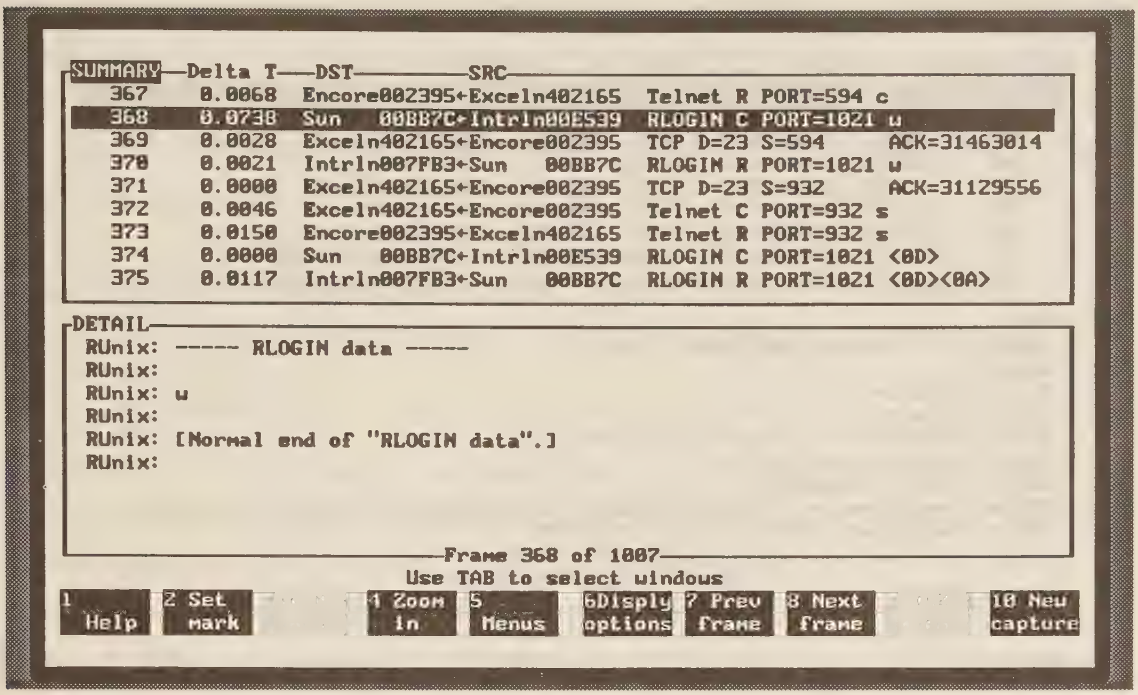
最初の行は，host1 の上のアカウントをもつ人みんなにこのシステムにアクセスすることを許していることを示している．2 行目のエントリは host2 上の user_a というユーザだけが信頼され，(host2 上の) その他の人は皆パスワードを示さなければならないということを表している．最後の 2 行では，group1 の全員を信頼して group2 の誰も信頼しないということを示している．グループは，netgroup データベース内で定義されている．

r* コマンドにおいてセキュリティの範囲を定める 2 番目の場所は，ユーザのホームディレクトリに存在する .rhostsファイルである．このファイルはhosts.equivファイルと同じ形式のものである．あるユーザのアクセス権をhosts.equiv中のエントリが拒否していても，そのユーザの .rhosts で認めていればそのユーザはアクセス権をもっていることになる．

最もよく使用されるコマンドを 3 つ挙げると，rsh (遠隔シェル)，rlogin (遠隔ログイン)，rcp (遠隔コピー) である．同様に使用されるコマンド (よく知られているものでは，プリントジョブ用の lpr)が他にもいくつか存在するが，それらについては 14 章で論じる．図 12-8 に rlogin コマンドの例を示す．rlogin のトラフィックと，同じサブネットワーク上の Telnet のトラフィックとが混ざっていることに注意しよう．

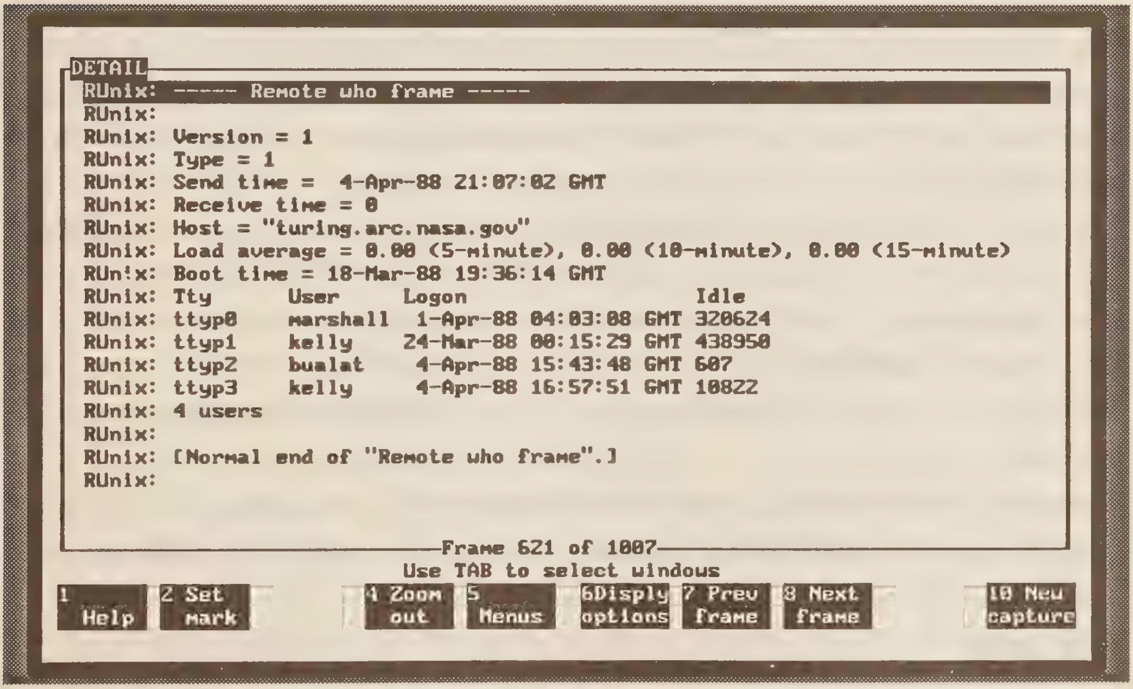
r* コマンドと同じ先祖に由来する他のコマンドが 1 つ存在するが，これはネットワーク管理者にそれほど馴染みがあるものではない．そのコマンドとは，rwho コマンドである．このコマンドを使う目的は，誰がネットワーク上にいるかを知ることである (14 章で論じる finger コマンドも同じような機能を与える)．

rwho コマンドは，行おうと試みていたことが原因ではなく，どのようにそれを行ったかが原因



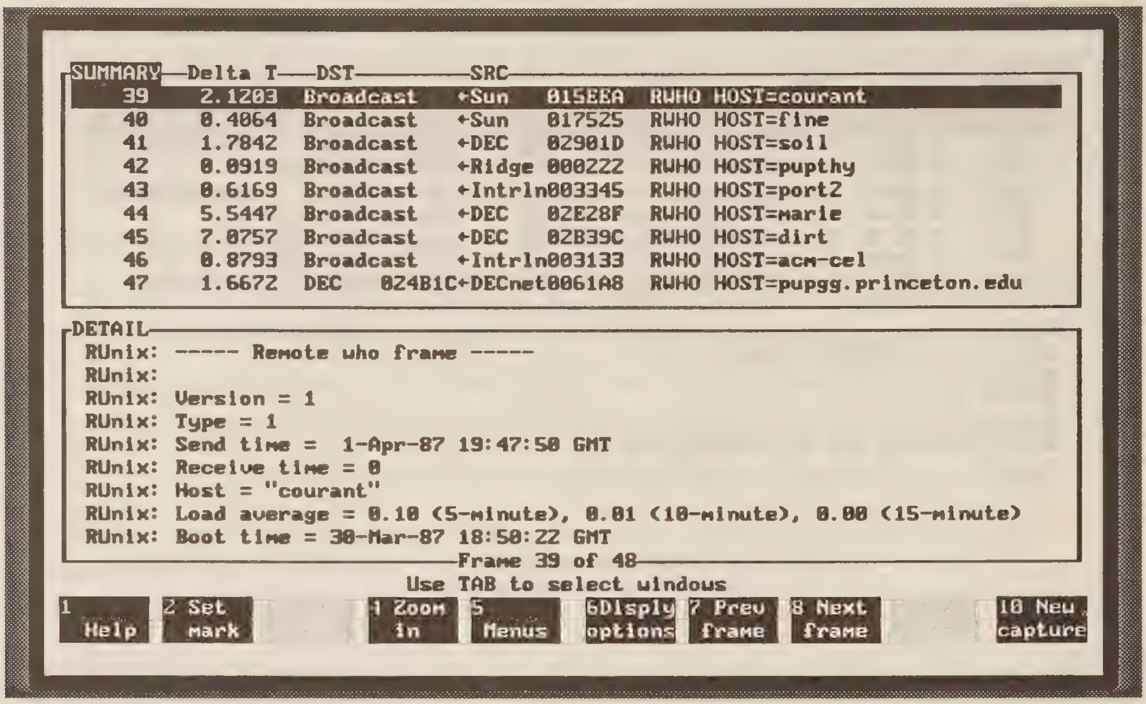
Courtesy of Network General

12-8 rloginのトラフィック



Courtesy of Network General

12-9 評判のよくないrwhoのプロトコル



Courtesy of Network General

12-10 理由のないrwhoのブロードキャスト動作

で悪い評判を受けた．図 12-9 と 12-10 に rwho コマンドの様子を示す．図 12-9 でわかるように，rwho コマンドは誰がログインしているか(さらに活動中のユーザと食事に出かけたユーザを区別するために，ユーザのセッション中で活動していない時間がどれくらい続いているか）に関する情報を与える．

図 12-10 は，rwho がどのようになし遂げられているかを示している．各ホストがこの情報をネットワーク上にブロードキャストしているのがわかるだろう．これは自己中心的なプロトコル設計として知られている．なぜなら，このプロトコルでは自分たちがログインすることにネットワーク上の他の人が実際に関心をもっていると仮定しているからである．多数のワークステーションがつながっているネットワークでは，rwho のトラフィック量は合計すると実際にかなりのものになり得る．

多くのネットワーク管理者は，rwho の代わりに finger を使っている．finger はユーザがある情報を要求するまで待っており，要求が来ると該当するホストに行って要求に対する答を見つける．

12.7 PC

PC-NFS は 2 つの意味でとらえることができる．狭い意味では，PC が NFS サーバを利用でき

るようにする機能である．しかし，そのパッケージ自体にはNFSそのものに加えて別の機能がいくつか含まれている．プリントサーバが含まれ，ネットワークベースのバックアップ機能とメールシステムもある．さらにPC-NFSプログラマツールキット（programmer toolkit：PTK）があり，これによってネットワークへのRPCインターフェイスをプログラマに与えている．Lifelineと呼ばれる追加パッケージが利用可能であり，これによってPC-NFSクライアントからUnixのメールにオンラインでのアクセスが可能になる．PC-NFSのユーザ（インターフェイス）部は，Microsoft Windows 3.0環境に統合されている．これはユーザがウィンドウ管理パネル内からネットワークベースのディスクドライブやアイコンを付加したり削除したりできることを意味する．

12.7.1 PC/NFS

NFSのPCバージョンの研究は，1985年の夏に始まった．その当時，NFSのすべての移植先は，基本的にバークレー4.2 BSD Unixの派生であった．多くの読者が覚えているように，1985年当時のDOSで提供されていたサービスはいわばまだ未発達であった．

PC/NFSエンジニアによる最初の評価では，PC用のTCP/IPの完全なバージョンは150kbytesのメモリを必要とした．DOSで利用できる総メモリは640kbytesであり，そのうちの30kbytesはconfig.sysファイルによって軽く使い尽くされてしまう．また，当時は終了後常駐機能（terminate and stay resident utility：TSR）を使用するのが一般的であり，それが少なくともさらに30kbytesのメモリを要したのである．

これで，あと430kbytesが利用可能な部分として残っている．これからDOSのコマンドプロセス（command.com）に必要な40kbytesを除くと，残るのは390kbytesである．さらに，WordStarやLotusのようなプログラムを走らせる必要がある．いうまでもなく，これらにはある種のデータも必要である．プログラムを走らせるには，少なくとも500kbytesのメモリを利用可能な部分として残しておくことが必要であるというのがPCの慣習的な知識である．

もちろん別の手段としてスマートアダプタ（smart adapter）に頼るというものがあった．これは，カードの上にTCP/IPを実装したイーサネットボードである．しかし，当時のほとんどのイーサネットコントローラはもっと原始的なものであり，スマートカードを利用してPC-NFSを実現することは，利用できる人が非常に限られてしまうことを意味したのである．

メモリは，NFSをPCに実装するときの多くの制約のまさに1つである．また，PCはマルチタスク能力をもっていないので，ノードが常にイーサネットポートにデータが来たかどうか監視していることを仮定しているネットワークアーキテクチャでは，NFSを実装するのは難しい．パケットがイーサネットに来ると，そのパケットはコンピュータによって処理されねばならないのだが，それができなければコントローラ上のバッファ空間はいっぱいになってしまい，パケットが失われ始めてしまう．

そこで，NFSを使用するためにはメモリの制約内でもちこたえなければならなかった．さらに，

マルチタスキングを行うためには，パケットが到着したかどうかを知るためにイーサネットカードを監視できるように，ソフトウェアはクロック要求を横取りしなければならなかった．市場の多くの TSR はクロック割込みを横取りして実現していたし，そのうちのいくつかはあまりうまく動作しなかったようである．いくつかの TSR を利用している場合，次の TSR に仕事ができるように割込みを渡すということをしないで，初めの TSR がメインプログラムに割込みを戻していた．

もう 1 つの実装上の問題はユーザの土台の違いである．VAX VMS と SUN Unix では，ユーザが定期的に調べる情報をたくさんもっている．その情報とは，オンラインマニュアルページ，チュートリアル(指導書)，そのコンピュータの指導プログラムといったものである．いいかえると，Unix ユーザはいくらかの時間を費やしてコンピュータがどのように動作するか学んだり，各自の特別な環境に合わせてソフトウェアを構成したりするものと考えられているのである．

しかし，PC の世界にはまったく異なったユーザの土台がある．ここでは，システムはあらかじめ設置されており，かなり簡単な方法で動作するものと考えられている．ほんのわずかなカスタマイズ（customization）と再構成（configuration）しか各 PC ユーザに期待してはいけない．この簡単な例を考えてみよう．ユーザが，Lotus 1-2-3 を使ってネットワークを介して蓄えられているスプレッドシート(spreadsheet)に操作をしたいとする．もし，サーバが利用できなくなるとどうなるであろうか？ “ドライブの準備ができていません（Drive Not Ready）”というエラーの代わりに，“サーバが利用できません（Server Unavailable）”というメッセージを Lotus が受け取れば，Lotus はきっとダウンしてしまうだろう．

12.7.2 PC-NFS アーキテクチャ

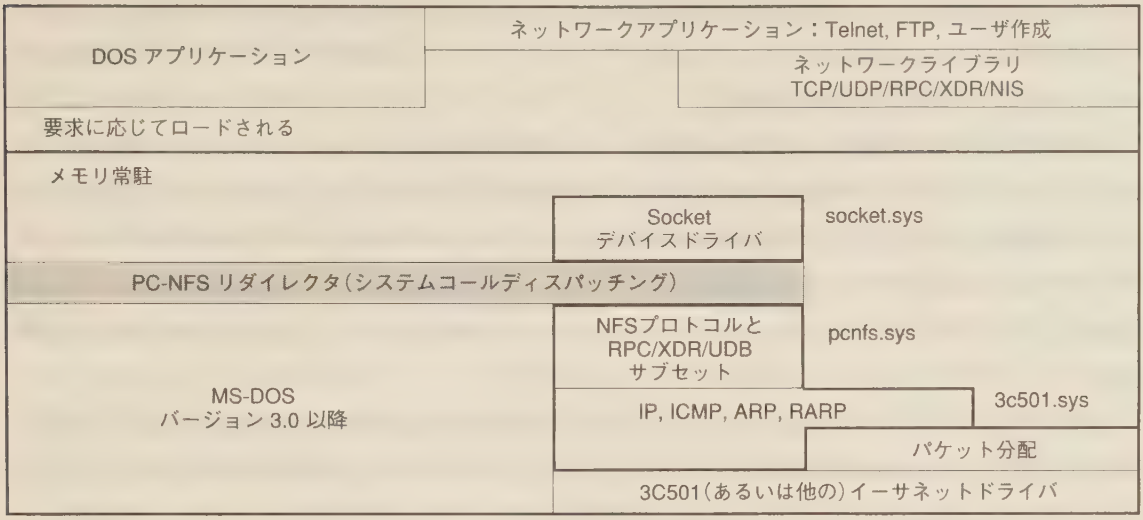
基本 NFS アーキテクチャには NFS クライアント，Telnet と FTP サービス，バークレーの遠隔シェルと遠隔コピーサービス，遠隔プリント，認証といったもののサポートが含まれる．PC はネットワーク上で最も遅いマシンであるので，少ないメモリからできる限りの性能をなんとか引き出すために，モジュール化した設計などはしばしば犠牲にされる．

このシステムは，3 つの異なる DOS ドライバから構成されている（図 12-11 参照）．

- pcnfs.sys ドライバ
- イーサネットコントローラドライバ
- socket.sys ドライバ

pcnsf.sys ドライバが，PC-NFS の仕事のうちの大部分を処理している．それには TCP/IP の IP と UDP や ICMP，ARP，RARP が含まれる．また，NFS とロックマネージャの処理を行うのに必要十分な RPC と XDR の最小のサブセットも含まれる．これが基本的に，ネットワーク層からプレゼンテーション層までの核となっている．完全な RPC サービスのような特別な機能は，アプリケーションプログラムにリンクされるライブラリによって処理される．

第 2 のドライバは，イーサネットドライバである．これは，もともとは最初の実装でサポートさ



12-11 PC-NFS アーキテクチャ

れた3Com イーサネットカードのモデル番号に由来する3c501.sys というものである．IP層はpcnfs.sys ドライバ上に実装されており，3c501.sys ドライバの送出ルーチンを呼び出していた．

パケットがイーサネットカードに受信されると，3c501.sys ドライバに渡される．ドライバはヘッダを読み取って，そのパケットを入力分配ルーチン（input demultiplexing routine）に渡す．分配操作は，通常 IP 層の機能であるが，効率化するためにイーサネット層に束ね込まれている．

分配コードは，パケットを受け取るのはどのクライアントであるかを識別する．各クライアントは，セレクタ（selector）と呼ばれるデータ構造によって識別される．このデータ構造は，入力データを保持しているバッファも指すことができる．セレクタが利用可能ならば，ドライバはデータをバッファにコピーしてクライアントを呼び出す．その後，クライアントはパケット処理を行う（さらに新しいバッファを割り付ける）ことができる．しかし，バッファが利用できないときはドライバはパケットを拾い落すように命令される．

セレクタには大きく2つのクラスがある．それはpcnfs.sys ライブラリ（例えばNFSやARP）の中にあるものと，ネットワークアプリケーションに使用されるものである．特別なネットワークアプリケーションには，3番目のドライバが必要である．それはsocket.sysである．socket.sysは，ドライバの支援機能を実装したものである．したがってread()やwrite()を用いるバークレー4.2 BSD のセマンティクスを利用できる．FTP はsocket.sys ライブラリを使用するプログラムの一例である．

ネットワークライブラリは，アプリケーションプログラムが実行可能ファイルにリンクする単なるファイルであり，これによってそのアプリケーションはTCP/IP やONC ルーチンに完全にアクセスできるようになる．このライブラリにはSun OS で見られる socket，RPC，XDR，NIS

ルーチンが含まれている．利用できないものの1つは，サーバ操作(例えばポートマッパ)をサポートするのに用いるRPCルーチンである．ライブラリルーチンはサービスを行うのに適切なデバイスドライバをそれぞれ呼び出す．

PCに伴う1つの興味深い設計上の問題は，イーサネットコントローラはまったく原始的なものであり得るということである．例えばもともとの3c501カードでは，1つ目のパケットをクライアントが利用できるようにするのと同時に別のパケットを受け取るということができなかったので，たて続けに来るパケットを受け取れなかった．カードが新しいデータを受け取れるようになる前に，クライアント（デバイスドライバ）はバッファ空間を空にしてしまわなければならなかった．

連続して届くパケットを受信できないことは，潜在的ではあるが現実の問題である．しばしばあることだが，NFSホストからデータがやってくるとそのデータはIP層で断片化されることになる．Unixの4096バイトのブロックは，最終的に少なくとも4つの別のイーサネットパケットに分割されることになるだろう．2番目の問題は，ブロードキャストの場合である．ブロードキャストがしばしば起こると，それを受け取ることは普通のデータの流れを妨げてしまうかもしれない．

TCPトランスポートのユーザには，単純な解決法がある．最大セグメント長(maximum segment size）を1024に設定して，すべてのデータを1つのパケットに制限してしまうことである．受信ウィンドウサイズ（recieve window size）を小さくしておくということは，ホストは同時に2つ以上のパケットを送ることができないということを意味する．NSFの場合，下位層としてUDPを使用しているので，解決法は(TCPの場合ほど)容易なものではない．NFSクライアント操作では，これは，大量の読取りは1キロバイトずつに分割されねばならないし，ディレクトリエントリの要求は一度には1キロバイトに制限されねばならないということを意味する．

問題は，UDPベースのアプリケーションが常に制御できるわけではないということである．公開されたファイルシステム中にはどのファイルシステムが含まれているかをたずねたら，どういうことが起きるだろうか？　マウントデーモンの方では，データのサイズに関して面倒を見ない．つまりリストを取り出し，それを送り出すだけである．もしリストがイーサネットの最大MTUより大きければ，断片化（fragmentation）が起こる．それは第2の断片（fragment）は失われることを意味し，さらにRPCがタイムアウトすることを意味する．もちろん解決法は，もっと知的な(intelligent)イーサネットカードを用いることである．歴史的な見通しからすると，ネットワーク機能をPCに組み込むために切り抜けねばならない障壁を理解しておくことは興味深いことである．

12.7.3　PC-NFSD

PC-NFSデーモン（PC-NFSD）は，標準のNFS操作の責任をもつだけでなく，2つの特別な

サービスの処理を行わなければならない．その2つとは，

・認証とファイル所有権/保護

・ローカルおよび遠隔プリントスプーリング

PC にはユーザやファイルアクセス管理の概念やマルチユーザシステムではおのずと決まっている他の点についての概念がないので，認証が必要である．ローカルおよび遠隔プリントスプーリングは，PC が遠隔プリンタにアクセスできるように追加されたサービスである．これによって PC-NFS は，サービスのレベルにおいてほとんどの PC ベースのローカルエリアネットワークと同等のものになっている．

PC-NFS における認証は数ステップで処理される．まず，クライアントはユーザ名とパスワードをデーモンに送らねばならない．すると，デーモンはローカルなオペレーティングシステムに対してこのユーザの認証を行う．認められればデーモンは Unix 形式の UID と GID をクライアントに返す．

セキュリティということになれば，この方法は決して安全なものではない．PC-NFSD に送られるユーザ名とパスワードは，偶然に拾い読みされるのを防ぐために符号化される．これは，決して破るのが困難であるように暗号化されるわけではない．

明らかに，これは世界で最も機密性の高い環境というわけではない．しかし，セキュリティと DOS は必ずしもうまく調和するというわけではない．セキュリティが要求されるところでは，SPARC ステーションや最近の他のワークステーションを使用するべきである．

PC-NFS デーモンの有用な特徴は，最近解決されたユーザ名はキャッシュできるということである．ユーザが同じユーザ名を共有している所で，名前とパスワードの表が非常に大きい場合にはこれは重要である．

ユーザ名をキャッシュする標準的な場合は，もちろん授業で使用するときである．この環境で各 PC に同じユーザ名を与えておくのは有用である．なぜなら，どの PC ワークステーションにアクセスしても，それを妨げるものがなくなるからである．

一般的にこのような環境においては，ほとんどの NFS ファイルは読出し専用ファイルシステムとして公開されている．PC 用のソフトウェアや読出し専用データはそこに保存される．典型的には，ユーザは実行結果用にローカルディスクを使うことを要求される．これは，あるユーザが他人のデータを見るのを防ぐためである．

12.7.4 ロック機構

PC 用のロック機構には，2つの問題がある．まず1つ目は，PC では Unix 環境とは違った種類のロックが利用できるということである．2番目は，優れた状態モニタを付け加えることは無理であるというのに充分な操作的な問題である．これらの両方に対処するために，ロックデーモンが強化された．さらに，追加的な機能をサポートできるように NFS デーモンが変更された．

PC-NFS にとって，レコードロック機構はきわめて重要である．もっと一般的な Unix 環境では，ロック機構は助言的なものであり，実際に使用しているアプリケーションはほとんどない(代わりに，ある特別な個別の解決法を用いている)．DOS においては，共有ファイルアクセスを管理する概念が与えられている．これは，Unix が無視しているものである．

ロックデーモンは PC 用の 2 つの拡張部をもつ．まず，デーモンは監視のないロック（unmonitored lock）への対処を行うことが必要である．次に，特別な RPC コールが DOS ロック機構のセマンティクスに付加された．監視のないロックを用いると，ロックを保持しているクライアントがまだ動作しているかどうかをサーバはチェックしないことになる．PC-NFS クライアントは，いずれかのファイルシステムをサーバにマウントしているときでも，すべてのロックを解放して，それによって前セッションのすべての面影を解放してしまう．PC はリブート時にロックの状態を保持しておらず，それゆえにロックを回復しようと試みないので，これは重要である．

サーバのリブートの間は，PC-NFS クライアントがロックの状態を保持している．PC は，サーバが回復するとロックデーモンとの接触を再確立する．ロックのセマンティクスが拡張されて，ファイルの共有とアクセスの概念はファイルロック機構と結合された．ファイルとレコードのロック機構は，Unix の SVID のロック機構とかなり似ている．ファイルアクセスは，他のユーザにある種のアクセスを許すべきかどうかを決定する第 2 の尺度である．これは，ロックデーモン（lockd）によって，特別なバイト 0 セマフォ（byte 0 semaphore）を用いて実現されている．

12.7.5　PC-NFSD の印刷機能

基本的な NFS の操作に加えて，PC-NFS デーモンはユーザが遠隔プリンタにファイルをスプールできるようにしている．クライアントは印刷ファイルをスプールしたいという意志を PC-NFSD サーバに伝える．サーバは，ファイルを書き込める NFS でマウントされたディレクトリのパスを返す．

この時点で，PC-NFS のアプリケーションはプリント要求を横取りし，そのデータをスプールファイルに書き込む．ユーザアプリケーションがデータを書き込み終わったら，PC-NFS クライアントはデーモンにファイルが印刷されるべきであることを知らせる．すると，デーモンは lpr のようなローカルな印刷機能を呼び出して，ファイルを印刷する．

より進んだ勉強のために

Sun Microsystems, *SunNet License Administrator's Guide*, Part No: 800-3307-10, Revision A of January 1990.

———, *SunNet License Production Tool User's Guide*, Part No: 800-3308-10, Revision A of 19 January 1990.

———, *SunNet License Production Tool Administrator's Guide*, Part No: 800-3496-10, Revision A of 19 January 1990.

Chapter 13

メール

13.1 概要

9章のドメイン名前システムの議論において，ドメインアドレスが与えられたときに，そのユーザに対するメールを処理する遠隔ホストの名前を，ローカルシステムがどのようにして見い出すのかを見た．それはMX(メール交換機の意)レコードであった．

ここでは，メッセージがどのようにして実際に転送されているのかを見る．Internetの世界では，単純メール転送プロトコル(Simple Mail Transfer Protocol : SMTP)というプロトコルが用いられている．SMTPは，単にメッセージ転送を行うものである．つまり，あるシステムから別のシステムにメッセージを移す手段である．

Internetでは，同一コンピュータ内に配達(もしくは転送，拒否)することは，ホストの何らかのプログラムにより行われる．SunのUnixシステムでは，これはsendmailというメール処理プログラムの領域である．他に利用できるメール処理プログラムもある．2つのハンドラーがSMTPで話せる限り，それらはメッセージを交換することができる．

メッセージを送る他の方法もいくつかある．厳密なUnixの環境下では，Unix間コピープログラム(Unix-To-Unix Copy program : UUCP)がよく使われる．sendmailというメール処理プログラムでは，メッセージを送るときUUCPとSMTPのどちらでも“話す”ことができる．

また，3番目のアプローチとして，X.400標準に基づいたものがある．そこでは，X.400のゲートウェイが用いられて，Unixに基づくシステムをより広いX.400のメッセージ環境に接続する．X.400は，ゲートウェイを用いて，多くの異なるローカルメールシステムを結び付ける接着剤としてしばしば用いられる．

13.2 SMTP

Internetの最初のアプリケーションの1つとして開発された，単純メール転送プロトコルから

始めることにしよう．皮肉なことに電子メールは，あとからの付け足しとして開発された．遠くにあるホストにログインできるようになることが，最も差し迫って必要なことであると考えられていたときに，メッセージを送ることが最も重要なアプリケーションになると思う人はほとんどいなかった．

しばらくの間，メッセージがどのようなものであるかという疑問に目をふさぐことにする．メッセージの受信者，表題，発信元，そしてメッセージの中身をどこに配置するか，メッセージの形式がどうなっているのかということは，メッセージの転送に関する限り関わりをもたないのである．メーラつまりリモートホストにメッセージを送る役を担っているローカルプロセスが必要とする情報を取り出す．通常これは，メッセージと誰行きのものであるかを記したファイルの名前だけである．

メーラが，他のホストにメッセージを送る必要があると判断すると，SMTP を話すプロセスをもつ周知のポートに対して TCP コネクションを開く．その後，送り手と受け手の間で対話が行われる．ここで，メッセージはまだ最終の目的地に到着していないかもしれないということを忘れてはいけない．つまり相手のシステムは，メールを受け取ってまた別のシステムに転送するメール交換機である可能性が大きい．

たとえメッセージが最終目的地に届いたとしても，相手のメーラがいかにして配達するのかは本当にはわからない．ほとんどのシステムは複数のユーザインターフェイスをもっている．例えば Sun の環境では，メールが配達されると旗が立つ小さな郵便受けを備えたウィンドウベースのアプリケーションと，コマンドラインで動かすユーティリティとをユーザは選択することができる．

SMTP の交換は，基本的には 3 つの段階から構成される．まず最初に，送り手が識別されなければならない．次に，そのメールの 1 人ないしそれ以上の受け手が識別される．各々の受け手に対して，相手側システムは，そのメールを受けることができるかどうかを判断する．つまり，受け手がその相手側システム上にいるか，またはその相手側システムがどのようにメールを別のホストへ転送したらよいかを知っているかどうかである．

最後に，実際のデータが送られる．たくさんの受け手を識別した場合でも，データは 1 コピーだけ送ればよいことに気をつけてほしい．たくさんの人に宛てられたメッセージの場合，これによって必要とするネットワーク帯域幅を劇的に少なくすることができる．

典型的な対話は次のような構成になっている．

```
MAIL FROM:<reverse-path>
250 OK
RCPT TO:<forward-path>
250 OK
DATA
```

345 INTERMEDIATE

これより後に送られるすべての行はデータであると見なされる.

データの終わりは終止符だけの行で表される.

250 OK

3つのコマンドのそれぞれに対する応答が，1つの数と1つのテキストメッセージから構成されている点に気をつけてほしい．数字はプログラムのためのものであり，それに対応するテキストは，たまたま見ている人間のための助言的なメッセージである．

3つの桁のそれぞれは応答の異なった局面を表している．最初の数字はコマンドが成功したかどうかを表している．例えば，2はコマンドが肯定的に完了したことを意味している．2番目の数字はメッセージが参照する対象を表している．5はメールシステムに関係する答であることを表している．3番目の数字はメッセージでもっと細かく指定することを可能にしている．このように250という応答は，メールシステムがコマンドを肯定的に完了したことを表している．

それでは，もう1つの対話についてもっと詳しく見てみよう．この例は，RFC821でのSMTPの記述の中にあるもので，図13-1に示されている．リモートシステムは，ローカルシステムがメールを送ろうとしているすべてのユーザに対するメッセージを処理することはできないことに注意してほしい．しかし，単に1つの受け手が拒否されたからといって，すべてのメッセージが拒否されたということにはならない．つまり，ある特定のユーザを見つけ出すことができないということは，メールシステムにおいてはよくあることなのである．ローカルシステムは，他のリモート

```
S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK

S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK

S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here

S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with
<CRLF>.<CRLF>
S: intermediate data
S: intermediate data
S: <CRLF>.<CRLF>
R: 250 OK
```

13-1 あるSMTPセッション

SMTPコマンド	
HELO	S:HELO USC-ISIF.ARPA R:250 BBN-UNIX.ARPA
MAIL FROM:<reverse-path>	
RCPT TO:<forward-path>	
DATA	
RSET	
SEND FROM:<reverse-path>	メール(端末に表示せよ)
SOML FROM:<reverse-path>	Send or Mail
SAML FROM:<reverse-path>	Send and Mail
VRFY<string>	メーリングリスト中の名前の確認
EXPN<string>	メーリングリストを展開する
HELP [space<string>]	
NOOP	無動作
QUIT	
TURN	役割の交代
SMTP アーキテクチャ上の制限	
ユーザ名	最大64文字
ドメイン名	最大64文字
Path (forward および reverse)	256文字
コマンド行	復帰や行送りを含めて512文字
応答行	上に同じ
テキスト行	1000文字
受信者バッファ	100受信者以内

13-2 SMTPコマンドとその制約

システムに対して，何度か試した後に，メッセージを配達することができないと判断する．

MAIL，RCPT，DATA といった基本的なコマンドに加えて，SMTP には他にいくつかのメッセージやオプションが定義されている(図 13-2 参照)．

これらのコマンドにより，メールボックスの名前を確認したり，誰がメーリングリストに属しているのかを見い出したり，そして 2 つのシステムの役割を変えることさえもできる．役割を変えるというのは，すべてのメッセージを送った後で，ローカルシステムがリモートシステムに対して相手のもっているメッセージの配達を認めるものである．

コマンドに対する答は，応答のコードとテキストで構成されている．テキストは助言的であり，人間が使うためのものである．これに対して数字はプログラムを導く．応答には 3 つの数字があり，各々の数字は異なる情報を伝達する(図 13-3 参照)．これと同じ応答コードの原理を，FTP プロトコルのところでまた見ることになるであろう．

最初の数字	
1	肯定的な予備応答:コマンドは受理したが，まだ実行していない．続行するかしないかを示す別のコマンドを送る．SMTPではこの応答を必要とするコマンドがないことに気をつけよう．
2	肯定的な完了応答
3	コマンドシーケンスグループに対して使われる肯定的な中間の応答
4	一時的な否定的完了:もう一度実行を要求できる
5	永続的な否定完了応答
2番目の数字	
0	文法
1	情報(状態，またはヘルプ要求に対する応答)
2	コネクション(伝送チャネルを参照する応答)
3	未定義
4	未定義
5	メールシステム
3番目の数字	
502	未定義の行動
504	コマンドは実装されているが，パラメータが実装されていない

13-3 SMTP リプライコード

13.2.1 ユーザ検索

リモートメーラはメッセージを配達することができなくても，ローカルメーラよりもメール転送をどう行ったらよいのかということについて，よりよい情報をもっていることがしばしばある．簡単な PC を例にとってみよう．PC はたくさんの異なるユーザから構成されるローカル領域に対してのメールハンドラとして機能することができる．通常，PC は何らかの形態でより大きなシステムをメール交換機として利用する．解決できないローカルメッセージはいずれも，常にそのメール交換機に送られる．メール交換機は，それらのメッセージを適切なネットワークに送る方法を知っている．

SMTP のリンクが PC とメール交換機の間で設定されると，PC はメールを最初の受け手に送ろうとする．SMTP ではメール交換機は 2 通りの異なる方法で答えることができる．

251 User not local ; will forward to＜フォワードパス＞

551 User not local ; please try＜フォワードパス＞

フォワードパス(forward path)は，単にメッセージを配達するために接続すべきホストのリストである．ローカルシステムは，フォワードパス中の最初のホストを見つけて(551 のメッセージの場合)，そのホストに送る必要がある．あるホストに送るためには，他に 2,3 のユーティリティが関わる必要がある．

ローカルメーラは望ましいホストをとり出して，ドメイン名前システムに“A”リソース，すな

わち目標のホストの Internet アドレスを調べるように要請を発する．IP アドレスが決まると，SMTP が用いる周知のサービスポートを通して TCP コネクション要求がそのホストに送られる．ホストにたどり着いたときに，そのホストがその特定の人に対するメッセージを実際に処理することができることを確かめるべきである．確認(verify : VRFY)コマンドにより，システムはユーザ名を送って，そのリモートシステム経由で，そのユーザにたどり着くことができることを確認するための応答を得ることができる．

もう 1 つの役に立つユーティリティとして展開(expand : EXPN)コマンドがある．もし，リモートユーザがメーリングリストである場合，リスト上に誰がいるのかを探し出すことは有用かもしれない．EXPN コマンドは文字列を受け取って，少なくともユーザ名を含み，場合によってはフルネームも含むような複数行の応答を返す．mconnect ユーティリティを用いてメールホストに接続し，SMTP の環境下でメールの実名やリストを展開することは珍しいことではない．これはユーザが Internet プロトコルと直接に会話する数少ない例の 1 つである．

簡単な VRFY コマンドと EXPN コマンドの例は，RFC821 に示す．

```
S : VRFY Smith
R : 250 Fred Smith＜　＞
S : VRFY Gouizenkyinplatz
R : 553 User ambiguous
S : EXPN Executive-Washroom-list
R : 550 Access Denied To You
```

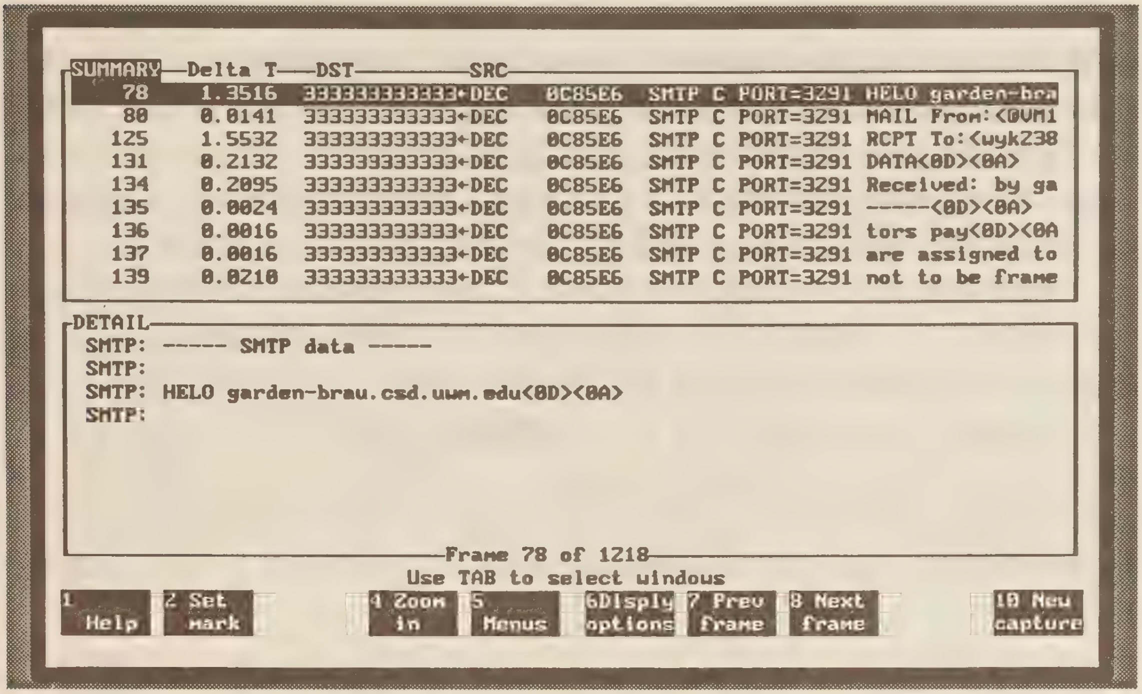
VRFY は，人の名前を受け付けるので，あるユーザのメールボックスを見い出す初歩的方法であることに注意してほしい．

もう 1 つ，あまり使われないが，ベルや呼び子の役割を果たすものとして SEND コマンドがあり，それはローカルメールボックスではなく，ユーザの端末上にメッセージを送るものである．Unix の talk ユーティリティが十分ではないとして，非常に打ち込んでいる(しかし，あまり生産的ではない)プログラマなら，TCP ソケットをオープンし，SMTP の SEND コマンドを送って，画面上にメッセージを表示するだろう．この風変りなコマンドの変形として，ユーザに対して send or mail(SOML)することや，send and mail(SAML)することすらできる．いうまでもなく，多くの場合 SMTP の仕事は DATA，MAIL，RCPT コマンドに限定されている．

13.2.2　経路

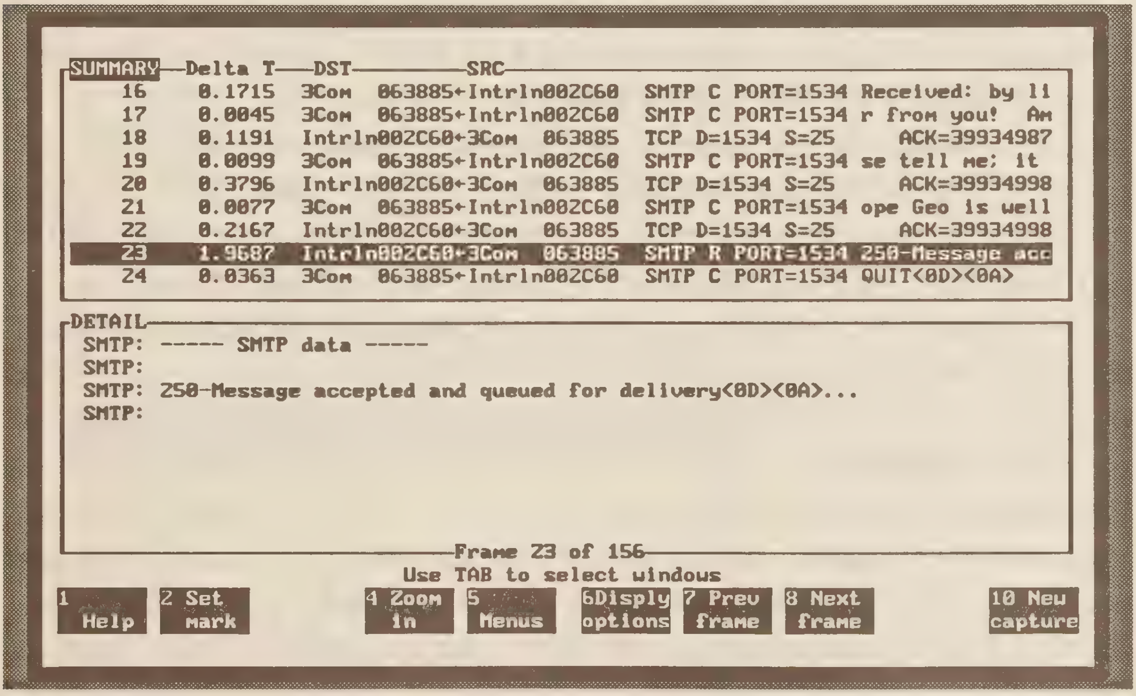
フォワードパスという言葉を何度か用いてきた．これはアドレスとまったく同じものではない．しばしばローカルデータベースが用いられて，あるメッセージを，あるパスから別のパスへと移動するのに使う必要のあるホストのリストが構成される．

ドメインアドレスの Joe@Three を例にとってみよう．ローカルデータベースを調べると，Three



13-4 SMTP セッション

Courtesy of Network General



13-5 SMTP セッション

Courtesy of Network General

というホストはアクセス不能のため(SMTPのポートをもっていないため)，直接たどりつくことはできないことがわかる．われわれはホストOneへはSMTPのパスを設定することができ，そして過去の経験からOneはメッセージをホストTwoに送り，そのホストTwoがメッセージを配達することができるということを知っている．

経路とアドレスの間には違いがある．あるホストがフォワードパスを受け取ると，そのホストは自分の名前を取り除き，送られてきた経路の中で次に指定されたホストはそのままにしておく．メッセージを受信したホストの名前が，受け取ったメッセージの最初にあるとは限らないことに気をつけてほしい．これは単に，ホストが指定されたホストに到着する方法を見つけるために転送用のデータベースを調べているのにすぎないことを示している．

図13-4と図13-5では，あるメールトラフィックの典型的な例を示している．図13-4では，HELOコマンドでセッションが初期化され，続いてメッセージが送られている．図13-5はTCPに基づく回線上のSMTPセッションを示している．データは下位のTCP層で確認されていて，SMTPのアプリケーションはデータが順序通りに送られることを仮定してよいことに注意されたい．

13.3 RFC822

SMTPは1つのメッセージをある地点から別の地点へ移す方法を定義している．知っていなければならないことは，メッセージが誰から来たのであり，誰に行くのかということだけである．メッセージ自体は，SMTPの関する限り単なる任意のデータである．

メッセージが内部的にどのようになっているのかはRFC822の領域である．RFC822はメッセージがどのようなヘッダをもち，本体はどのような形をしているのかを議論している．RFC822はテキストメッセージの形式だけを定めている．この拡張が本体部分のいろいろな種類に関してIETFで論じられているものの，RFC822はマルチメディアのメールには適していない．

アドレスを識別することに加えて，ヘッダフィールドは機械と人間の両方に役立つような広く様々な目的に使われている．例えば，ほとんどのメッセージはメッセージIDフィールドをもっている．あるメッセージに対して返事が送られると，もとのメッセージIDを付した“In-Reply-To”ヘッダフィールドが挿入され，リモートユーザインターフェイスが対話の中で異なるメッセージを組み合わせることができるようになっている．

われわれの多くにとっては，そのようなヘッダフィールドは役に立たない．もし，単純で原始的なユーザインターフェイスをもっているのなら，メッセージIDを見ても，それを無視して表題のフィールドにすぐ目が移るであろう．一方，意匠をこらしたユーザインターフェイスをもっているのであれば，そのプログラムは気が利いていて，データベースの中からIn-Reply-Toのヘッダで参照されている古いメッセージを引き出してきて，それを新しいメッセージとともにスクリ

ーン上に表示するかもしれない．

RFC822 ではごく少数の必須ヘッダしか定義していない(図 13-6 参照)．残りはすべて省略可能である．すべてのヘッダがあったとして，それらを用いて何かするかどうかを決めるのはユーザインターフェイス次第である．例えば，Receipt-Requested ヘッダを挿入することができる．しかしながら，多くのメールシステムは受領書を作成しない．つまり，誰もその機能をプログラムし

フィールド名	記述
Return-path:	返信アドレス
Received:	
Received from:	ドメイン（送信ホスト）
Received by:	ドメイン（受信ホスト）
Received via:	物理経路
Received with:	リンク/メール プロトコル
Received ID:	メッセージ番号
Received for:	アドレス（本来のアドレス指定） 受信したフィールドはすべて最後に “;date-time” をもつことができる。
Reply-To:	アドレス
From:	
Sender:	
From:	
Resent-Reply-To:	
Resent-From:	
Resent-Sender:	
Resent-Date:	
Date:	
To:	
Resent-To:	
CC:	
Resent-cc:	
Bcc:	見えない複製 (Blind carbon copy)
Resent-bcc:	
オプションフィールド	
Message-ID:	
Resent-Message-ID:	
In-Reply-To:	
References:	
Keywords:	
Subject:	
Comments:	
Encypted:	

13-6 RFC 822 ヘッダフィールド

ていない．“私はあなたのメッセージを受け取りました．”というメッセージを送り返すことは人間に任されている．多くのメールインターフェイスは，メールメッセージ中のほとんどの制御フィールドを無視する（表示しない）ように設定することができる．

メッセージのヘッダフィールドは厳格に定められた形式をもっている．それらはコロンで終了する文字の列であるフィールド名から構成されている．コロンの後には，各々のフィールドに固有なデータがある．基本的な RFC では，いくつかのヘッダフィールドが定義されている．ユーザは独自のヘッダフィールドを付け加えることができる（技術的には，フィールド名は X-Fruit-of-the-Day のように X-で始まるべきである）．さらに他のいくつかの標準が，それ以外のヘッダフィールドを定めている．例えば，プライバシー強化メール(Privacy Enhanced Mail)は，メッセージの完全性検査や信任状，その他のセキュリティ情報を，どのようにしてメッセージのヘッダに保持するかを定義している．

拡張フィールド（これらは RFC822 への公式な拡張として発表されたもの）もあり，これらはどれも X-で始まる名前をもっていない．ユーザ定義のフィールドも，公表されていない拡張であり，公表された拡張によって置き換えられる可能性がある．

13.4 転送

メッセージがあるホストから別のホストへと転送されるごとに，いろいろなトレースフィールドがメッセージに付け加えられる．これらのフィールドのほとんどは，メッセージの復路を定めるのに役立つ．発信者と戻りの経路について明確な情報をもつことは重要である．復路とは発信者への戻りの経路のことである．ちなみに発信元経路指定が提供するものより都合のよいアドレスをユーザが指定したり，返事を取り扱うための別のユーザ名を入れておくための Reply-To というフィールドもある．

Received フィールドは，各々のトランスポートのサービスによりトレースとして付け加えられる．メッセージを取り継ぐ各トランスポートサービスが Received ヘッダを入れる．Via と With というパラメータを付け加えることもできる．実際には，どのプロトコルが使われているのかを十分表現するために，いくつかの With パラメータを用いることがある．

図 13-7 は，典型的なメールメッセージの例を示している．メッセージが転送されるにつれて付けられた多数の Received ヘッダに注目しよう．最初の転送には SMTP が使われているが，2 番目には別のプロトコルの UUCP が使われていることに気をつけよう．この例では，Sun OS の Vacation コマンドに相当するプログラムが自動的に応答したものである．きちんとした Vacation プログラムは，分配リストに対しては自動応答を送ろうとしない．しかし，あるユーザに対するメーリングリストによる配達のすべてが，リスト全体に対する返答の嵐という結果を引き起こすことがあることはよく知られている．

```
From hqafsc-vax.af.mil!postmaster'%hqeis.mrgate."a1.decnet
            Fri Dec 21 19:36:39 1990 remote from uunet
Received: by malamud.com (UUL1.3#5653)
           from uunet with UUCP; Fri, 21 Dec 90 16:51:04 PST
Received: from HQAFSC-VAX.AF.MIL by uunet.uu.net (5.61/1.14)
            with SMTP id AA13352; Fri, 21 Dec 90 19:36:39 -0500
Message-Id: <9012220036.AA13352@uunet.uu.net>
Date: 21 Dec 90 19:32:00 EST
From: ``HQEIS::MRGATE::''A1::POSTMASTER'"
           <uunet!hqafsc-vax.af.mil!postmaster'%hqeis.mrgate."a1.decnet>
Subject: Automatic reply to mail addressed to MCDANIEL
To: ``carl'' <carl@malamud.com>

From:      NAME: Mail Postmaster
   FUNC:
   TEL:                          <POSTMASTER AT A1 AT HQEIS>
To:        WINS%"CARL@MALAMUD.COM"@KRDEV@MRGATE


THE AFSC DDN PROGRAM MANAGER DUTY STATUS.

I will be on leave from 14 Dec 90 till 21 Dec 90.

I will return to duty on 24 Dec 90.

Emergency or Urgent DDN & AF Concentrator issues can be addressed
to:  Hq AFSC/SCXR
     Major McCoy
     DSN 858-7904
     Coml: 301-981-7904
     Email Address: MCCOY@HQAFSC-VAX.AF:MIL

Routine issues will be handled upon my return, so please leave a
message or forward an email message.

NOTE:  THIS IS AN COMPUTER SYSTEM AUTO REPLY MESSAGE AND IS SENT
OUT EVEN TO MAILER DISTRIBUTION LISTS, TCP-IP, INFO-VAX, AND
OWNER-IETF EMAIL MESSAGES.

Regards,
```

13-7 自動的な応答生成

もう 1 つ，興味あるメッセージが図 13-8 にある．ドメインからドメインへとメッセージが移動するたびに，何度もこのメッセージが転送されていることに注意しよう．このメッセージは，Usenet ネットワーク上のニュース掲示として始まったものである．メッセージはバークレーの ucbvax に到着し，そこで ISO 開発環境 (ISODE) のメーリングリストに宛てられたメールメッセ

```
From NIC.DDN.MIL!isode-RELAY  Wed Nov 21 20:58:15 1990
          remote from uunet
Received: by malamud.com (UUL1.3#5653)
          from uunet with UUCP; Wed, 21 Nov 90 18:07:59 PST
Received: from NIC.DDN.MIL by uunet.uu.net (5.61/1.14)
          with SMTP id AA09280; Wed, 21 Nov 90 20:58:15 -0500
Received: from ucbvax.Berkeley.EDU by NIC.DDN.MIL
          with TCP; Mon, 19 Nov 90 16:51:49 PST
Received: by ucbvax.Berkeley.EDU (5.63/1.42)
          id AA04161; Mon, 19 Nov 90 16:50:07 -0800
Received: from USENET by ucbvax.Berkeley.EDU with netnews
          for isode@nic.ddn.mil (isode@nic.ddn.mil)
          (contact usenet@ucbvax.Berkeley.EDU if you have questions)
Date: 13 Nov 90 07:31:21 GMT
From: uunet!ucsd.edu!usc!zaphod.mps.ohiostate.edu!mips!cs.uore-
gon.edu !logicse!zephyr.ens.tek.com!tektronix!nosun!qi-
clab!m2xenix!quagga!ucthpx!uctcs.uucp!gram  (Graham Wheeler)
Organization: Dept. of Computer Science, University of Cape Town
Subject: ISODE - what is it; where can I get it?
Message-Id: <887@ucthpx.UUCP>
Sender: uunet!nic.ddn.mil!isode-relay
To: isode@nic.ddn.mil

OK, I guess I'm dumb. All this time I've been killing articles on ISODE
'cos I haven't known what it was and haven't been too interested. Sud-
denly I put 2 and 2 together, and figured 'dev-environ == DE' Aha!

So now I'm interested, but I can't go back and read all the news I've
killed. So at the risk of boring everyone, can someone give me a brief
description of ISODE, and also tell me if it is possible to get it (from a
mail server; I can't ftp from here). To spare others, please E-mail a re-
sponse. Thanks.

Graham Wheeler                          | "Don't bother me,
Data Network Architectures Lab          | I'm reading a `Crisis'!"
Dept. of Computer Science
                    | Internet <gram.uctcs@f4.n494.z5.fidonet.org>
University of Cape Town
                    | BANG: <...uunet!ddsw1!olsa99!uctcs!gram>
```

13-8 メールメッセージ

ージに変えられた.

そのメッセージは，それからネットワーク情報センターへ送られる．そこには実際のISODEのメーリングリストがある．Isode@Nic.DDN.Milは他と同様なユーザであり，適切なホストとInternet アドレスを見い出すためにDNS の MX レコードが調べられる．TCP コネクションが設

定されて，SMTP のようなプロトコルが用いられてメッセージが転送される．

メールメッセージが Isode@Nic.DDN.Mil でメーリングリストプログラムに送られると，そこにあるプログラムはそれらのメッセージを別の発信元アドレスの isode-relay に送る．relay からメッセージが来るようにするということは，プログラムが自動的にメーリングリストデータを見つけることができるということ（そして自動返答を生成しないということ）を意味する．

isode-relay からメッセージは分割され，別々のパスに送られる．この特定のメッセージはヴァージニアにある商用メール取り継ぎサービスである UUnet と呼ばれるマシンに行った．UUnet から malamud.com というマシンに行き，そしてユーザ Carl に到着した．

もう 1 つ面白い点として，逆戻りの経路を再構成するのに用いられたメッセージ上の from 行に気をつけよう．この UUCP 形成のアドレスは，特定のユーザに到達するのに使われることのできる経路を示している．経路の中で，各ホストは !(“bang”) 文字によって他と区切られている．

13.5 From，Sender，Reply-To

メッセージの送り主が，実際にそのメッセージに責任をもっている人ではない，というような例がいくつかある．典型的な例は，秘書が自分のアカウントでメッセージを送るときである．このとき，2 つの別の付加フィールドが役に立つ．Sender フィールドはメッセージを打ったのが誰であるのかを示し，Reply-To フィールドは誰が返事を処理するのかを示す．

```
From : Geory Jones＜Group@Host＞
Sender : Secy@host
Reply-To : Secy@host
```

これらのフィールドのもう 1 つの使い方として，メッセージを送るのは 1 人だが，その返事は多くの異なる人に行くべき場合がある．この場合，Reply-To フィールドには何人もの人が含まれていることになる．

```
From : George Jones＜Jones@host. Net＞
Sender : Jones@Host
Reply-To : The Committee : Jones@Host.Net,
                           Smith@other.org,
                           Doe@Somewhere-Else;
```

13.6 メッセージのカプセル化

普通，メッセージのテキストは単にそれだけのものであり，プログラムに対して何も意味をなさない(そして，プログラムを使っている人間に対してさえも無意味かもしれない)一連の ASCII

文字である．

いくつかのメッセージを1つにまとめる必要がある場合がある．そのようなときには，伝送のためにメッセージを他のメッセージの内側にカプセルづめにするのである．いくつかのメーリングリストは，とても大きな量のトラフィックを生み出す．例として，Sun-Spots メーリングリストがあり，それは何百もの人に送られる(したがって，何百ものメッセージを生み出す)．各々のメッセージを各人に送り出すと，信じられないほどの量のトラフィックを発生することになる．

その代わりに，Sun-Spots リストはダイジェスト形式を用いている．多くの異なるメッセージをすべて束ねあげて1つの大きなメッセージ(典型的には，15 000 から 18 000 バイト)にする．ダイジェストが届けられると，ユーザはそれを切りとって，それから個々のメッセージに対して返事したいと思うであろう．

とても簡単な慣習が用いられている．まず初めに大きなパッケージを示す通常のヘッダ部がある．そのパッケージの内側には，カプセル化境界(EB)によって分離された一連のメッセージがある．ここで，カプセル化境界とは単にダッシュからなる行である．

もし，メッセージの中に実際にダッシュからなる行をもっているなら(または行の先頭にダッシュがあるなら)，その最初の文字が単純にダッシュとスペースの組に変えられる．メッセージを切りとるときに，ダッシュとスペースで始まっている行は，カプセル化の境界として扱われない．単純だって？　もちろんその通り．それがすべてなのだから．

13.7 メッセージの転送に対する DNS の使用法

読者は，9章でのドメイン名前システムのリソースレコードの議論を思い出すかもしれない．MX レコードは特別な種類のリソースレコードであり，それはメール転送のために広く用いられている．

MX レコードは，与えられたドメイン名に対して，どのシステムがメール交換機として振る舞うかを示している．それに加えて優先度の値がある．1つのドメインに対して，いくつかの異なる MX レコードが存在することができる．

優先度の数字は，メーラが特定のホストに配達するのに使う順番である．最も低い数字の MX が，最初に試すべきものである．同じ優先度をもつ複数のレコードもあり得る．

メーラが使用することを選択するリソースレコード(RR)が，他にも少しあるということに気をつけよう．正規名(CNAME) RR は，問い合わせられたドメイン名は，実際には他のドメイン名に対する別名であることを表している．

周知のサービス(well-known service : WKS) RR は，与えられたドメイン名がサポートするネットワークサービス(例えば SMTP)に関する情報を蓄積している．最後に，メール交換機として働こうとしているあるホストが与えられると，われわれはもちろん，アドレス(A)レコードを調べ

て，そのホストの Internet アドレスを見い出す必要がある．

メッセージのルーティングにおいて最も必要とされることは，メールがループするのを防ぐことである．宛先のホストに対して MX としてリストされているホスト上のメーラは，より低い優先度の値をもつ MX にしか配達してはいけない．

もう 1 つの潜在的な問題は，DNS からの不完全な回答である．もし，回答が不完全であるとすると，最も重要な MX RR を見逃すことになるかもしれない．それゆえ，メーラは完全な答をもった応答だけを受け取る．このことはトランケーションビットがセットされている答を受け取ったメーラは，バーチャルサーキットを使って問合せを続けるということを意味する．

第 1 段階は，メーラが相手のドメイン名に関する MX RR を求める問合せを発することである．もし，問合せが成功すれば(トランケーションビットがなくエラーコードもない)，メーラは次の 2 つのリソースレコードのうちの 1 つを受け取るだろう．

- MX RR のリスト
- CNAME RR

もし，CNAME が戻ってきた場合には，MX レコードのリストが提示されるまで問合せが繰り返される．メーラが MX RR のリストを受け取ると，メーラはこの特定のメールを処理するのに適切なホストを決定しようとする．もしリストが空であれば，メーラは MX がリモートホスト自身であるかのように扱う．

それ以外の場合，メーラは優先度が最も低いものを見て，そのドメインが望ましいメールサービスをサポートしているかどうかを知るために WKS の問合せを発する．この問合せは IP アドレスとポート番号を返し，メーラはそこで SMTP セッションを確立しようする．

メーラがリストをたどるときに，望ましいサービスをサポートしていない RR を，(WKS フィールドに基づいて)取り除くということに注意されたい．もし，メーラが RR のリスト中にあれば，優先度の値のより大きな他のホストをすべて取り除く．

これらをすべて取り除いた後，メーラはまた空のリストでに行きつくかもしれない．これは誰も望ましいメールサービスをサポートしていないために起こったのかも知れない．あるいはドメイン内のシステムが，このメーラがハンドラだと思っているためかも知れないが，実際にはそうではない．いずれの場合にしろ，メッセージは配達することができない．

13.8 UUCP

SMTP は，メール転送プロトコルの一例である．他によく用いられているものとして，Unix 間コピープログラム(Unix to Unix Copy Program : UUCP)がある．UUCP は，バークレーの Unix が SMTP をサポートする以前からのものであり，ダイヤルアップの要求駆動コネクションに今でも使われている．UUCP は，単にファイルを転送すること以上のことができる．例えば，ある Unix

コマンド	記述
cu	コンピュータを離れたコンピュータに接続する．最初の回線を落とすことなしに，ファイル転送を行ったり片方のコンピュータ上でコマンドの実行ができる．
uucp	ユーザが，あるコンピュータから別のコンピュータへファイルをコピーできるようにする．UUCPはワークファイルとデータファイルを作り，転送のためのジョブを待ち行列にして，uucicoデーモンを呼び出す．
uuto	あるコンピュータから別のコンピュータの共用のスプールディレクトリ(/var/spool/uucppublic/receive)にファイルをコピーする．UUCPはリモートコンピュータ上で利用者がアクセスできるどんなディレクトリに対してもコピーできるようにする．uutoはリモートユーザにuupickで，それを取り出すように告げる．
uupick	/var/spool/uucppublic/receiveに置かれた(uutoにより転送された)ファイルをユーザに取り出させる．
uux	リモートコンピュータ上でコマンドを実行するのに必要なワークファイルとデータファイルを作成する．そして，リモートコンピュータ上で実行すべきコマンド文字列を含む第3のファイルすなわち実行ファイルを付け加える．
uustat	uucp，uuto，uux等で起動された転送要求の状態を表示する．待ち行列中の転送要求を制御することができる．
管理用のコマンド	
uulog	uucp，uuto，uuxの利用記録をもつログファイルの中身を表示する．
uucleanup	スプールディレクトリをきれいにする．
uutry	uucicoデーモンを起動することで，呼出しのテストをする．
uucheck	ディレクトリ，プログラム，ファイルの状態が正しいことを確認し，明らかな文法誤りを探し出す
UUCPデーモン	
uucico	回線として用いるデバイスを選択し，回線を確立し，ログインして，データや実行ファイルを転送し，結果のログを取って，転送の完了をメールでユーザに知らせる．2つのuucpデーモンが互いに通信し合う．
uuxqt	リモート実行要求を実行する．リモートコンピュータから送られてきた実行ファイルをスプールディレクトリから探す．それをオープンしてデータファイルのリストを得る．ファイルが利用可能かどうかを調べ，これを行うことが認められているかどうかパーミッションファイルを調べて，実行する．
uusched	スプールディレクトリの待ち行列中の仕事をスケジュールする．まず，リモートコンピュータが呼び出す順番をランダムにし，それからuucicoを起動する．

13-9　UUCPコンポーネント

システム上のユーザが，別のシステムにある仕事をしてもらって，結果を返してもらうための，ワークオーダーを送ることができる．

UUCP は，実はたくさんのプログラムを集めたものである (図 13-9 参照)．システムは少し不可解だが，広く実現されているという長所がある．さらに詳しい情報に関しては，O'Relly and Asociate によって書かれた 2 つのうすい本を大いに薦める (この章の終わりに掲載してある)．

UUCP は電話線やケーブルのようなポイント対ポイントの回線上や，イーサネットのようなネットワークの上で動作する (LAN や WAN の環境では，SMTP メーラが用いられることの方があ

りそうだが).様々なプログラムによって,ユーザが他のシステムに接続したり,ファイルを移動したり,データを受け取ったりすることが可能となっている.それに加えて,リモートシステムは何者であるか,それらにダイヤルする方法,それらのシステムが利用者のファイルシステム上で何ができるのか,どれだけ頻繁にそれらのシステムを呼び出したらよいかといった情報が様々なデータベースに含まれている.

このぞんざいな書き方から,著者が UUCP をいくらか嫌悪感をもって見ていることがわかるだろう.しかしながら,多くの古いシステムと同様に,それはとても重要な目的を果たしている.何年もの間,Internet は研究機関と大きな政府の官僚主義により厳しく限定されてきた.

政府と契約関係のない人達もメールメッセージを交換する方法を必要としていた.UUCP はこの目的を果たし,今でも果たしている.著者は,自分の PC を UUnet の商用メール交換サービスに接続する方法として UUCP を利用している.

UUCP は,もう 1 つの貴重なサービスである Usenet の基盤を形成している.Usenet は,とてもゆるく組織化された掲示板とニュースのサービスである.基本の考えは,コンピュータが別のコンピュータとメッセージを交換することに同意することである.そのコンピュータが,また別のコンピュータとメッセージ交換することに同意するのである.

正式な経路づけがなく非厳格なメッシュなので,ときどき国の片側から反対側へメッセージが移動するのに何日もかかってしまうことがある.これと同時に,国の反対側の人々が,彼ら自身のメッセージを送り出している.このことは各々のノードが掲示板の状態を,少しずつ異なって見ているということを意味している.

このシステムはとても大きな長所をもっている.それは略式であることだ.多くのシステムで,ローカルにメールを分配したり,Internet に接続するのに TCP/IP と SMTP を使う一方で,ニュースやメッセージに対しては UUCP の回線を使い続けている.

Sun のメーラである sendmail は,トランスポートメカニズムとして SMTP と UUCP の両方で(ユーザが特に付け加えたそれ以外のメカニズムも同様)メッセージを送ることができる.sendmail は必要な宛先アドレスに従って,どのメールトランスポートを使うのかを指図するためのいくつかのルーティングテーブルをもっている.

13.9 メッセージの munging

いくつか異なるメッセージ処理環境がある.厳格でない UUCP が 1 つの例であり,Internet の RFC822 が別の一例である.他に MCI Mail,Compuserve や X.400 等がある.各々の環境は,異なるメッセージフォーマットをもっている.

メッセージを,ある環境から別のものへと移動することは munging と呼ばれている.システムがゲートウェイを経由して,どんどん一緒に接続されていくにつれて munging が次第に問題とな

ってきている．電子メールは，ほとんどいつでも最もよく用いられるアプリケーションであり，異なる環境の境界をまたいで動作する必要がある．

かつて munging はかなり簡単であった．システムは異なるアドレスの形式ではあるが，本質的には同じ種類のテキストの移動メッセージを取り扱うことができればよかったのである．これはほとんどのヘッダが無視されていたためである．つまり，送り手と受け手のような情報だけが明示的な動作を必要としていたのである．

簡単な変換の例は，Internet のメッセージを UUCP ドメインに移す場合である．UUCP では，メールアドレスは感嘆符によって分離された明示的な経路により構成されている．

UUnet!Malamud!Carl

このアドレスは，メッセージをホスト UUnet からホスト Malamud へ，そして最後にユーザの Carl に移動することを示している．ところが，よい Internet アドレスは次のようなものである．

Carl@Malamud.Com

DNS サーバで Carl@Malamud.Com を探すことにより，ホスト UUnet が Malamud.Com に対する MX のメール交換機であるという事実がわかるであろう．

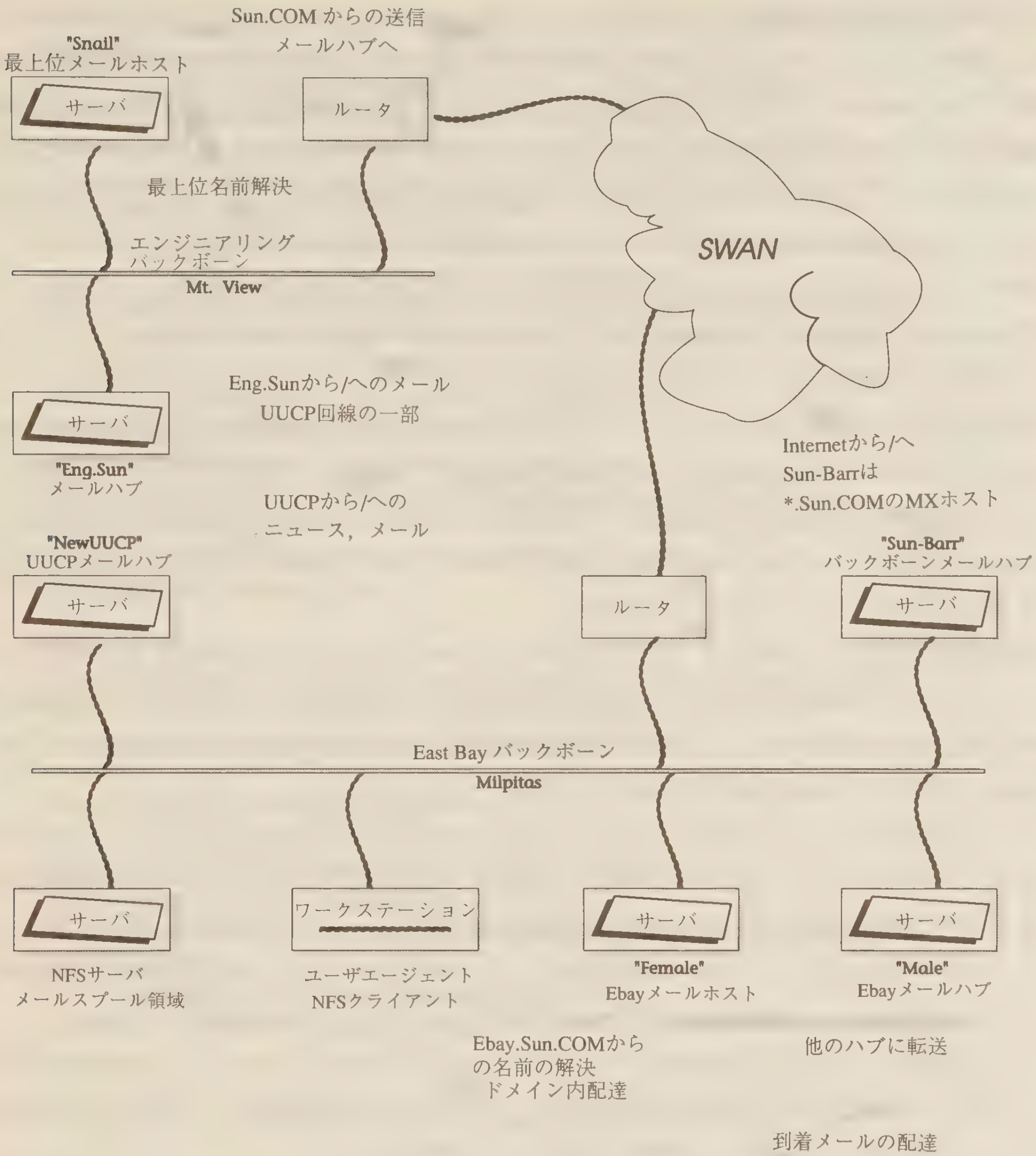
この情報があれば，われわれは簡単に DNS のドメインアドレスを UUnet のアドレスに変えることができる．UUnet はメール交換機であるから，それを最初にもってくる．次に Malamud を付け加え，最後が Carl となる．X.400 の世界に行ったり来たりする場合には，もっと複雑な munging が必要である．これは X.400 では様々なサービス要素が利用可能であるためである．ある特定の動作を表すヘッダ行がなくてはならないのである．

13.10 事例研究：Sun メールシステム

X.400 の中身に入る前に，実際のメール処理システムがどのように構成されているかを知ることは有益であろう．図 13-10 は，Sun の EastBay(Milpitas) サブドメインのメールマシンの高レベルの概要を示している．読者も知っての通り，メッセージをあちこちに移動させるにはいろいろな方法があり，異なるメッセージ処理ドメインの間の接続がある．ワークステーション上のユーザは，ウィンドウベースのメールツールのような何らかのユーザインターフェイスプログラムをもっている．メールツールはメールファイルを見い出すのに特定の領域，典型的には配達領域とユーザのホームディレクトリにある格納領域を見るように設定されている．

おそらくそのメールは NFS サーバであるホストの上にあるのであろう．そこがメールメッセージが最終的に配達される場所である．これらの場所は Milpitas サブドメイン内に散らばっており，これらはすべて“Male”と“Female”と呼ばれる 2 つのホストと通信し合っている．

Male は EastBay 用のメールハブであり，Female は EastBay ドメイン用のメールホストである．その他のドメインも同様のサーバをもっている．メールホストは，外に出ていくすべてのメール



13-10　Sun メールサーバ

メッセージについて責任をもっている.

メールホストである Female は名前(宛名)の解決を行う. あるユーザがメッセージを“Jack”に宛てたとすると, それはおそらく EBay ドメイン内のメッセージであるので(Jack@EBay), Female は名前を解決し, Jack のメッセージが保管されている領域を見つけ, メッセージを配達する.

一方, 例えば Jill@Eng のようにメールがドメインの外に宛てられている場合には, Female は

その名前を可能な限り解決し，そのメッセージがSunのエンジニアリングドメイン宛であるとわかる．他のメッセージではこうも容易にはわからないかもしれないが，Femaleはできる限りの解決をした後，Maleにメッセージを転送する．Maleはメールハブであり，すべてのデータを最終目的地に向けて一歩前方に送り出す．そのメッセージが，UUCP経由で送出される場合には，2つのUUCPハブ，すなわちNewUUCPまたはEngSunのいずれか1つに行くことになる．これらのマシンは，世界中のいろいろなホストとUUCPに基づく関係を保っている．

他のメッセージでは，Internet経由で出ていくことになる．Internetに出ていくすべてのメールは，Sun-Barrというホストで処理される．もし，あるユーザがInternetからSunドメインの中にメールを送った場合も，そのメールはこのホストに行く．

Sun-Barrは，Sunの内部ネットワークと外部のInternetへの中継点である．例えば，Jack@Sun..Com宛にInternetの他の地点(例えばOtherCompany.Com)から送るとしよう．その相手方のシステムは，ドメイン名前システムを用いてSun.ComのMXレコードを検索する．これによりホストSun-Barr.Sun.ComとそのIPアドレスが示されよう．そこでSMTPセッションが設定され，メールが転送されることになる．

Sun-Barrではローカルデータベースが調べられ，Jackなる人は誰であるかを見い出す．おそらくメッセージは，Jack@EBay.Sun.Com宛であることがわかるであろう．実際には，この時点ではSun.Comの部分は余分であり，必要な情報はJackがEastBayにいることだけであり，メッセージはMaleに送られてJackへ最終的に配達されることになる．

もう1つ注目すべきホストがあり，それはSnailである．SnailはSunの広域ネットワークにおけるトップレベルのメールホストである．このホストは解決できないメッセージが最後の試みとして送られる場所である．ネットワークが外部世界に対していくつかの異なるリンクをもっており，メッセージを正しいメールハブに転送することが必要なので，このような機能をは重要である．図13-11には，この階層を上に向かって運ばれたメッセージの例が示されている．

これらのホストはすべてsendmailというメールハンドラを走らせている．sendmailはメッセージを見て，次に受け取るべき相手を決定する．メッセージが宛先に着いたときには，sendmailはメッセージをしかるべきスプール領域に入れることで配達を行う．

メッセージがまだ宛先にたどりついていない場合には，メッセージは分析されて，次の受け取り手が決められる．sendmailにおけるメーラとは，共通の配達特徴をもつ相手先メールハブのクラスのことである．

図13-12には，Sunのネットワークにおける主要な4つのsendmailプログラムの統計量が示されている．この統計量はメーラごとに分割されていることに注意されたい．特定のメーラの行うべきことはsendmail.confというファイルで定義されているが，このsendmail.confの文法は政府の契約のようにはっきりしないものである．特定のメーラについての完全な記述に関しては設定ファイルの詳細な調査を必要とするが，ある程度の一般化を行うことはできる．MaleとFemale

```
From EBay.Sun.COM!terryb  Thu Jan 24 13:45:13 1991 remote from uunet
Received: by malamud.com (UUL1.3#5653)
          from uunet with UUCP; Thu, 24 Jan 91 12:54:30 PST
Received: from Sun.COM by uunet.uu.net (5.61/1.14)
          with SMTP id AA17654; Thu, 24 Jan 91 13:45:13 -0500
Received: from EBay.Sun.COM (male.EBay.Sun.COM) by Sun.COM
(4.1/SMI-4.1) id AA07745; Thu, 24 Jan 91 10:45:10 PST
Received: from female.EBay.Sun.COM by EBay.Sun.COM (4.1/SMI-4.1)
          id AA26176; Thu, 24 Jan 91 10:45:06 PST
Received: from seescandy.EBay.Sun.COM by female.EBay.Sun.COM
          (4.1/SMI-4.1-900117) id AA09681; Thu, 24 Jan 91 10:36:45 PST
Received: by seescandy.EBay.Sun.COM (4.1/SMI-4.1)
          id AA02897; Thu, 24 Jan 91 10:43:08 PST
Date: Thu, 24 Jan 91 10:43:08 PST
From: uunet!EBay.Sun.COM!terryb (Terry Beyer)
Message-Id: <9101241843.AA02897@seescandy.EBay.Sun.COM>
To: carl@malamud.com
Subject: Mainframe Printing on Sun Printers Summary
Cc: terryb@EBay.Sun.COM

Carl -
  The attached is a sample weekly summary of the volumes printed
from our mainframes on to Sun based printers.
  Please let me know if you have questions.
  Terry
—— Begin Included Message ——

From sunprint@irca8 Wed Jan  2 13:58:29 1991
Date: Wed, 2 Jan 91 14:00:31 PST
To: martyto@mock, proman@doubleplay, rf@zeppelin, robg@alps, so-
lan@sunnyskies,
      stevek@wonder, terryb@seescandy
Subject: SUNPRINT Weekly Summary


                    SUNPRINT WEEKLY SUMMARY

      From Previous Wed 14:00 to Wed Jan  2 14:00:04 PST 1991

 PRINTER        REQUESTS     AVG BYTES     MIN BYTES     MAX BYTES
```

13-11 Sun EBay 領域からのメッセージ

は似たようなメーラを定義しており，メッセージは(NFS に基づく単純なコピーを用いて)自ホスト内に配達されるか外に送出される．

SunUUCP というメーラは，UUCP プロトコルを経由して外に出ていくトラフィックを処理しており，メッセージを Sun.Eng または NewUUCP へ転送する．Relay というメーラは，自ホスト

Mailer	Msgs From	Bytes From	Msgs To	Bytes To
Mail Statistics for Female				
Mail statistics from Tue Jan 22 00:02:16 1991 to Tue Jan 22 23:59:48 1991				
0 local	127	526K	96	2K
2 ether	12122	76413K	25139	100439K
5 relay	909	3795K	31788	144097K
9 sunuucp	2	2K	1303	7929K
Total	13160	80736K	58326	252467K
Mail Statistics for Male				
Mail statistics from Tue Jan 22 00:00:23 1991 to Tue Jan 22 23:59:44 1991				
0 local	6078	39803K	434	19K
2 ether	1886	15473K	17995	63529K
5 relay	8059	41480K	38387	157190K
9 sunuucp	2	3K	1291	7946K
Total	16025	96759K	58107	228684K
Mail Statistics for Snail				
Mail statistics from Tue Jan 22 00:00:06 1991 to Wed Jan 23 00:00:03 1991				
0 local	600	2004K	1948	3193K
1 prog	17	184K		
2 ether	771	3340K	120060	536758K
4 sales	137	1318K	804	6210K
5 swan	14	58K	143	562K
6 newswan	11081	62335K	15847	68819K
8 internet	2246	8758K	3259	23552K
Total	14849	77813K	142078	639278K
Mail Statistics for Sun-Barr				
Mail statistics from Tue Jan 22 04:05:06 1991 to Wed Jan 23 04:04:57 1991				
0 local	223	863K	2	2K
2 ether	8	25K		
3 ddn	11454	61641K	19628	93185K
4 fakeuucp	20	106K	5	17K
5 sales	135	540K	838	3344K
6 relay	21	46K	4092	29690K
7 area	4011	31713K	5825	17868K
8 uucp	588	2122K	1191	5504K
10 badarea	1952	14941K	2992	12507K
11 unknown	68	163K	2989	8586K
Total	18472	112135K	37570	170728K

13-12 Sunメーラの統計データ

内で解決あるいは配達できなかったメッセージのことであり，これらのメッセージはホスト Snail に送られ，そこでどのように処置すべきか判断される．Ether はイーサネットの範囲内での配送を処理するメーラである．

ホスト Snail と Sun-Barr は，ハブとしての中心的な役割から，はるかに複雑なメーラをもっている．Sales というメーラは，UUCP に基づくセールスオフィスへのリンクを管理する必要性から

発生した特殊なメーラである．そしてDDNは外に出ていくメッセージを処理するためのメーラである．

Sun-Barrにおける2つの興味深いメーラはBadareaとUnknownである．Badareaは誤った形式の宛先のためのものである．多くの人は，ホスト名をアドレスの中に書き込んでしまう．例えば，Jack@eng301c.Eng.Sun.Comというように．このようにホスト名を書き込んでしまってもメッセージは通るが，潜在的な問題をもつものである．例えば，あるユーザが他のホストに移動した

UUtraffic for NewUUCP									
Remote Host	Kbytes			Hours			Avg CPS		
	Recv	Sent	Total	Recv	Sent	Total	Recv	Sent	Total
varsun	23249.4	4685.2	27934.6	17.9	3.4	362	385	2014	1088
lsil	8295.1	14422.4	22717.5	2.6	2.9	870	1395	2622	5348
emsca	2601.3	11634.7	14236.0	7.8	29.0	93	111	2278	3554
teamone	30.5	11390.0	11420.5	0.0	14.3	175	222	36	152
enera	5950.7	3098.5	9049.2	3.7	3.4	449	255	210	1246
valid	693.9	7882.8	8576.7	0.3	1.5	555	1466	438	3290
xopusw	5654.8	2129.8	7784.6	3.3	0.4	476	1513	1746	546
cadence	2876.0	3404.9	6281.0	0.8	0.8	955	1203	258	246
sunisv	451.4	3425.2	3876.6	0.6	4.3	202	220	194	2119
iis	65.4	2423.1	2488.5	0.2	5.8	82	116	112	369
svmsg	1583.8	375.7	1959.6	2.6	0.8	168	125	648	198
ntrlink	7.2	1207.7	1214.9	0.0	0.2	571	1724	4	923
sunuru	22.4	1109.7	1132.1	0.0	1.8	144	172	30	158
poseur	316.0	520.7	836.7	0.2	0.1	461	1663	262	313
larse	32.4	802.1	834.5	0.0	1.0	191	220	14	24
synergy	7.0	474.2	481.2	0.0	1.2	91	110	12	14
soland	60.5	390.2	450.6	0.8	3.9	21	28	58	160
ufrt	156.1	265.8	421.8	0.5	0.5	88	155	228	290
wesfdd2	35.3	332.1	367.5	0.1	0.4	183	251	32	196
barcode	165.1	53.0	218.1	0.1	0.0	388	1795	116	48

SUMMARY	
Total active uucp sites:	31
Total files recv:	11,426
Total files sent:	20,632
Total files:	32,058
Total hours recv:	41.8
Total hours sent:	76.3
Total hours:	118.1
Total Kbytes recv:	52,399
Total Kbytes sent:	70,586
Total Kbytes:	122,985

13-13 "NewUUCP"のUUCP統計データ

としたら何が起こるだろうか.

Badarea ハンドラ中の名前は解決できるものであるが，不可解なアドレスが含まれている場合もある. 外から入ってきたあるメッセージが Jack@Sun.Com 宛であるが, Jack というユーザは誰であるか見当もつかない. あるいは, あるメッセージが Bob@NonHost.TheMoon 宛に出ていこうとするとき，表を探してもトップレベルのドメインである TheMoon に到達する方法を見つけることはできない. そこで DNS の問合せを発行することになる. 結局，このドメインへのハンドラを見い出す手段をもたないことがわかり，このメッセージは/dev/null へ送られる.

NewUUCP という機械は多種の異なるホストへのリンクを保持している. これらのホストの多くは, Sun のセールスオフィスであったり, 顧客であったり, あるいは単にメッセージやニュースの交換のために Sun と契約を結んでいる組織であったりする. 図 13-13 に，このホストにおける UUtrafic コマンドの一例を示す.

13.11 X.400

X.400 は，一連のメッセージ伝搬エージェント（message transfer agent）とユーザエージェントを管理ドメインの中に集めた構成，すなわち資源の集合を一元管理する構成となっている. 国際通信連合（International Telecommunication Union : ITU）のメンバー（または登録されたプライベートメンバー）により管理されているドメインは Administration Management Domain (ADMD) と呼ばれる. 他のドメインはすべて Private Management Domain (PRMD) である.

PRMD は複数の ADMD と接続されているかもしれないが，ある 1 つのメッセージに関しては, PRMD は 1 つだけの ADMD に接続される. PRMD におけるルーティングとは, メッセージを ADMD に送ることである.

ADMD におけるルーティングに関しては，直接接続されているドメインの場合は簡単である. それより先へのルーティングを行おうとする場合, その手順は仕様の外である. しかしここで OSI のセション，トランスポート，ネットワークの各層を用いれば，ドメインは容易に他のドメインに“直接”接続され得ることに注意すべきである. 当該ドメインのアドレスを見い出すためには X.500 のディレクトリが用いられる.

X.400 のドメインはメッセージ伝搬エージェントとユーザエージェントから構成されている. メッセージ伝搬エージェントが組み合わさってメッセージ処理システムが構成される. そしてさらにメッセージ格納領域 (message store)，アクセスユニット (access unit)，分配リスト (distribution list) といった特別な構成要素がある.

メッセージ格納領域は，メッセージ伝搬システムからメッセージを拾い上げ，ユーザのためにそれらを保持する単なるステージング (staging) 地点である. ユーザはメッセージを直接メッセージ伝搬システムに送り出すこともできるし, メッセージ格納領域に保管させておくこともできる.

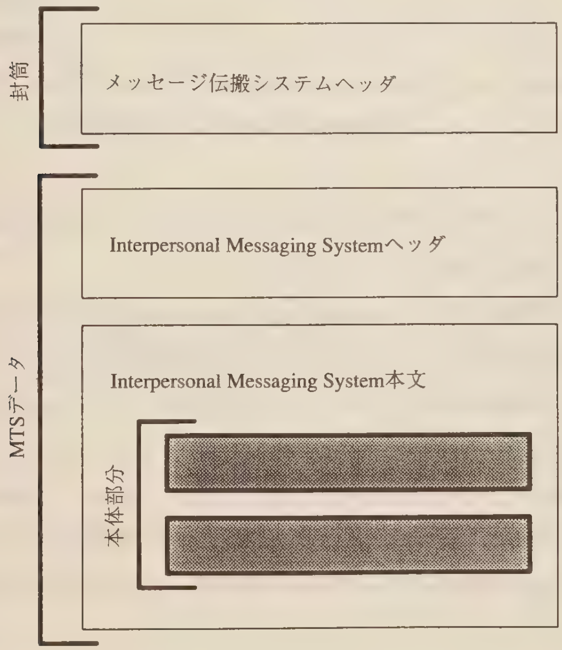
分配リストはアドレスの組であり，それら各々がメッセージに置き換えられる．アクセスユニットは他のメッセージ処理環境に対するゲートウェイである．X.400は，テレックス，郵便，その他いくつかの環境との間の相互接続のためのアクセスユニットを定義している．DECのX.400ゲートウェイはアクセスユニットの例である．

メッセージ格納領域は，メッセージ伝搬エージェントにとってはユーザエージェントとして見える．すなわちメッセージ格納領域は，配達されたメッセージを受け取り，ユーザのために保持しておく．さらにユーザはメッセージ格納領域にメッセージを差し出し，その後メッセージ格納領域がそれらをメッセージ伝搬エージェントに渡すといったことも可能である．PC（ユーザエージェント）がVAXを郵便局（メッセージ格納領域）として用いる場合などがこのシナリオの例である．メッセージ格納領域は，その名の通り単なる記憶装置でもよい．ユーザのためのメッセージ自動転送といった付加サービスをメッセージ格納領域の内に作りつけることもできる．

13.11.1 メッセージ

X.400はメッセージを3つの部分に分解する（図13-14参照）．各メッセージにはメッセージ伝搬システムにより付加されたヘッダが付いている．それ以外の部分はメッセージ伝搬システムにとってはデータとみなされる．

Interpersonal Messaging Service（IPMS）と互換性のあるようなクラスのユーザエージェントは，メッセージをさらに構造化することができる．上位レベルプロトコルと同じように，この層はさらにヘッダと本文（body）から構成される．1つのIPMSのメッセージは，いくつかの異なる本体部



13-14 IPMSメッセージの構成要素

分(body part)をもつことができる．それぞれの本体部分を異なる形式で構造化することも可能である（例として，1つの部分は通常の ASCII テキストであり，別の部分はグラフィックイメージであるようなものがあげられる)．いくつかの本体形式を1つのメッセージの中に含められることは，マルチメディアメールを直接サポートしており，X.400の強力な点の1つといえる．

13.11.2 名前づけ

X.400 サービスのユーザは，その送信者/受信者(originator/recipient：O/R)アドレスとして知られる．この場合ユーザは個人のユーザであるかもしれないし，分配リストであるかもしれない．またX.500のディレクトリ名を使うことも可能であり，この場合X.400のアドレスが返される．

ネットワーク層のアドレスづけと同様に，目的としていることは個々のユーザに重複のないアドレスを与えることである．OSI のネットワークアドレシング規格と同様に，いろいろと異なる形式が定義されている（図 13-15 参照）．4種類の異なるアドレス形式が定義されている．それぞれについて様々な要素がユーザを識別することができる．いくつかの項目は必須の(mandatory)ものであり，他のものは条件付き(conditional)，すなわち使用するかどうかはそれぞれのドメイン次第というものである．例えば郵便の宛先を考えてみると，国名と郵便番号は必須の(required)項目である．これに対し物理的な配達サービスの名前は省略可能(optional)である．例えば米国の場合には，この情報は Federal Express，UPS，HDL もしくは米国郵政事業(United States Postal Service)などを表すであろう．しかし配達サービスが独占されている国では，この欄は必要ない．

13.12 IPMS

Interpersonal Messaging Service(IPMS)は，ユーザエージェントのサブクラスである．IPMS は特定の構造をもったメッセージを定義する．メッセージは勝手な(arbitrary)ものではなく，ヘッダといくつかの本体部分から構成される．

アドレスフォーマットと同様に，IPMS にも様々な項目(element)がある．いくつかのものは基本項目と考えられ，すべての場合に実装される（図 13-16 参照)．

基本サービスとしては，例えば内容物の種類の表示(content type indication)などがある．内容物の種類の表示は，IPMS のヘッダに記載されるフィールドであり，内容物がどのようなものであるかを表す．また各種のタイムスタンプも用意されている．そしてユーザがオプションのサービスを利用できるかどうかを登録できなければならない．

メッセージの構造に加えて，IPMS にはいくつかの通知が含まれている．メッセージ伝搬サービスは，配達，配達不能の通知を発行する．IPMS はこれに加えて，受信，未受信通知を含めることができる(もちろん IPMS を提供しているユーザエージェントが，未受信，受信の通知を発行しないこともできる)．

属性値	O/Rアドレスフォーム			
	記号	数字	郵便	端末
一般				
管理ドメイン名	M	M	M	C
共通名	C			
国名	M	M	M	C
ネットワークアドレス				M
数字使用識別子		M		
機関名	C			
機関ユニット名	C			
個人名	C			
プライベートドメイン名	C	C	C	C
端末識別子				C
郵便ルーティング				
物理的配達サービス			C	
物理的配達国名			M	
郵便番号			M	
郵便アドレシング				
拡張郵便アドレスO/Rアドレス構成要素			C	
拡張物理的配達アドレス構成要素			C	
ローカル郵便属性値			C	
物理的配達局名			C	
物理的配達局番号			C	
物理的配達機関名			C	
物理的配達個人名			C	
私書箱アドレス			C	
局留アドレス			C	
通りアドレス			C	
非形式郵便アドレス			C	
一意の郵便アドレス			C	
ドメイン定義済み	C	C		C
M-必須の C-条件付きの				
注：郵便アドレスは形式化されていない(unformatted)ことがある（すべての情報が1つの属性値で表される）．この表は形式化された郵便アドレスに適用される．形式化されていないアドレスは，「形式化されていない郵便アドレス」という属性値からなる．				
出典：CCITT Blue Book Rec.X.402(p.121)				

13-15 X.400のアドレス

基本サービス	アクセス管理
	内容物の種類の表示
	変換された表示
	配達タイムスタンプ表示
	IPメッセージ識別子
	メッセージ識別子
	配達不能通知
	もともとの符号化された情報種類表示
	投函タイムスタンプ表示
	種類分けされた本文
	ユーザ/UAケーパビリティの登録
出典：CCITT Blue Book VIII.7-X.400(p.38)	

13-16 基本的なIPMSの構成要素

IPMS には多くの異なるオプション項目が定義されている．これらのオプションは，もしサポートされていれば IPMS のヘッダのフィールドとして通常構成されている．図 13-17 に提供されている様々なサービスを示す．これらのサービスは，メッセージの送信側に提供されるものと受信側に提供されるものとに分類されている．各カテゴリーのそれぞれに関して，重要項目であるか (essential)，付加項目であるか (additional) にさらに分類されている．重要項目は IPMS により提供されなければならないが，ユーザはこれを選択しなくてもよいものである．付加項目は提供するかどうかの決定が個々の実装（もしくは管理ドメイン）に委ねられているものである．

付加項目の簡単な例としては，物理表現の付加 (additional physical rendition) があげられる．これは，ユーザがメッセージのコピーを印刷することを許すものである．この項目は両方の側で省略可能である．また自動転送表示 (auto-forwarded indication) は，メッセージの送信側に対して付加項目である例である．すなわち，送ったメッセージが他のユーザに自動転送されたとしても，必ずしもこれを送信側に通知する必要はないからである．これに対し，この項目は受信側にとって重要項目と考えられる．つまりメッセージを受け取ったユーザが，メッセージが実際に自動転送されたのかどうかを知ることができなければならないからである．

いくつかの項目はメッセージごとに利用することはできず，あらかじめ契約しなければならない (図 13-18 参照)．例えば，メッセージ伝搬システムはメッセージを受け取ってもそれをしばらくそのままにしておき，あとになってから配達することができる．暗黙の変換 (implicit conversion)，再方向づけ (redirection)，限定配達 (restricted delivery)，その他この類のサービスは，メッセージ伝搬システム内に余分な資源を必要とする．

X.400 のメッセージ環境の豊富さを示すために IPMS において可能な項目の（ほとんど）全部を含む一覧表を示した．すべてのユーザインターフェイスが，これらすべての項目をどのように扱うのかを知っているわけではないという点に注意されたい．下位 IPMS がこれらの機能をサポ

項目	送信側	受信側
物理表現の付加	A	A
代理受信許可	A	A
権限ユーザ表示	A	E
自動転送表示	A	E
基本的な物理媒体による表現	A	E
ブラインドコピー受信者表示	A	E
本体部分暗号化表示	A	E
内容物の機密性	A	A
内容物の完全性	A	A
変換禁止	E	E
情報が失われる場合の変換禁止		N/A
カウンタ収集	A	E
アドバイス付きカウンタ収集	A	A
相互参照表示	A	E
遅延配達	E	N/A
遅延配達の取消し	A	N/A
配達通知	E	N/A
Bureaufaxサービスによる配達	A	A
ディレクトリ名による受信者の指定	A	N/A
他の受信者の公表	A	E
分配リスト展開表示	N/A	E
分配リスト展開禁止	A	A
速達メールサービス	A	E
有効期限表示	A	E
明示的な変換	A	N/A
転送IPメッセージ表示	A	E
配達等級の選択	E	E
重要性表示	A	E
不完全コピー表示	A	A
言語表示	A	E
最後の配達指定	A	N/A
メッセージフロー機密性	A	N/A
メッセージ出所証明	A	A
メッセージセキュリティラベリング	A	A
メッセージ順序完全性	A	A
複数宛先配達	E	N/A
複数の本体部分	A	E
未受信通知要求表示	A	E

13-17 オプションのIPMSサービス

項目	送信側	受信側
配達の非拒絶	A	A
出所の非拒絶	A	A
投函の非拒絶	A	A
旧式表示	A	E
通常メール	A	E
送信者表示	E	E
送信者による代理受信者要求	A	N/A
MHSによる物理的配達通知	A	A
PDSによる物理的配達通知	A	E
物理的転送の許可	A	E
配達不能通知の防止	A	N/A
主要とコピーの受信者表示	E	E
プローブ	A	N/A
プローブ出所証明	A	A
配達証明	A	A
投函証明	A	A
受信通知要求表示	A	A
送信者による再方向づけ不許可	A	N/A
書留メール	A	A
人間としての受信者宛の書留メール	A	A
応答要求表示	A	E
応答IPメッセージ表示	E	E
報告の出所証明	A	A
転送アドレスの要求	A	A
要求された配達方法	E	N/A
内容物の返還	A	N/A
機密度表示	A	E
特別配達	A	E
保存メッセージの消去	N/A	E
保存メッセージの取出し	N/A	E
保存メッセージの一覧	N/A	E
保存メッセージの要約	N/A	E
表題表示	E	E
配達不能メールの返還	A	E
分配リストの使用		A
出典：CCITT Blue Book VIII.7-X.400		

13-17（続き）　オプションのIPMSサービス

代理受信者割り当て
配達の待機
暗黙の変換
入ってくるメッセージの再方向づけ
限定配達
セキュアアクセス管理
保存メッセージの警告
保存メッセージの自動転送

13-18 IPMSの契約機能

ートしていても，ユーザインターフェイスは付加項目を無視する場合が多い．このことから，ユーザはときおりメッセージを単なるASCIIエディタで検証し，付加項目が含まれているがユーザには見えないのかどうかを確かめることができる．

13.13 メッセージ伝搬エージェント

メッセージ伝搬システムには，次の3種類のオブジェクトが存在する．

・メッセージ

・プローブ

・報告

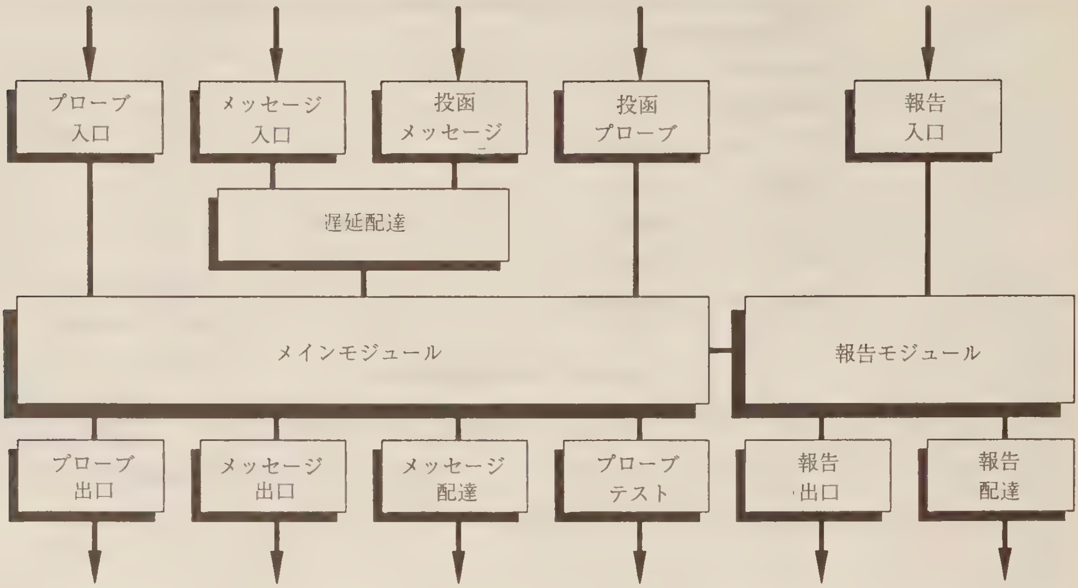
プローブは，あるユーザから他のユーザの直前のメッセージ伝搬エージェントまで伝えられるものであり，接続性を確かめるために用いられる．報告は，メッセージ伝搬システムからユーザに送られるオブジェクトである．各々の報告はメッセージやプローブに関するものである．

図13-19と図13-20に，メッセージ伝搬エージェントのモジュールの構成を示す．入ってくるメッセージはメインモジュールか報告モジュールへと送られ，その結果，メッセージや報告やプローブが送り出される．

メインモジュールは様々な機能を提供する．例えば分割(splitting)は，複数の受信者をもつメッセージをいくつかのメッセージに分割する．この分割は，1人の受信者がある方向，他の受信者が別の方向にいる場合に実行される．また結合(joining)は省略可能なステップで，同じ場所へ向かういくつかのメッセージを結合させる．例えば，1つの分配リストから出た10の配達不能メッセージは結合され得る．

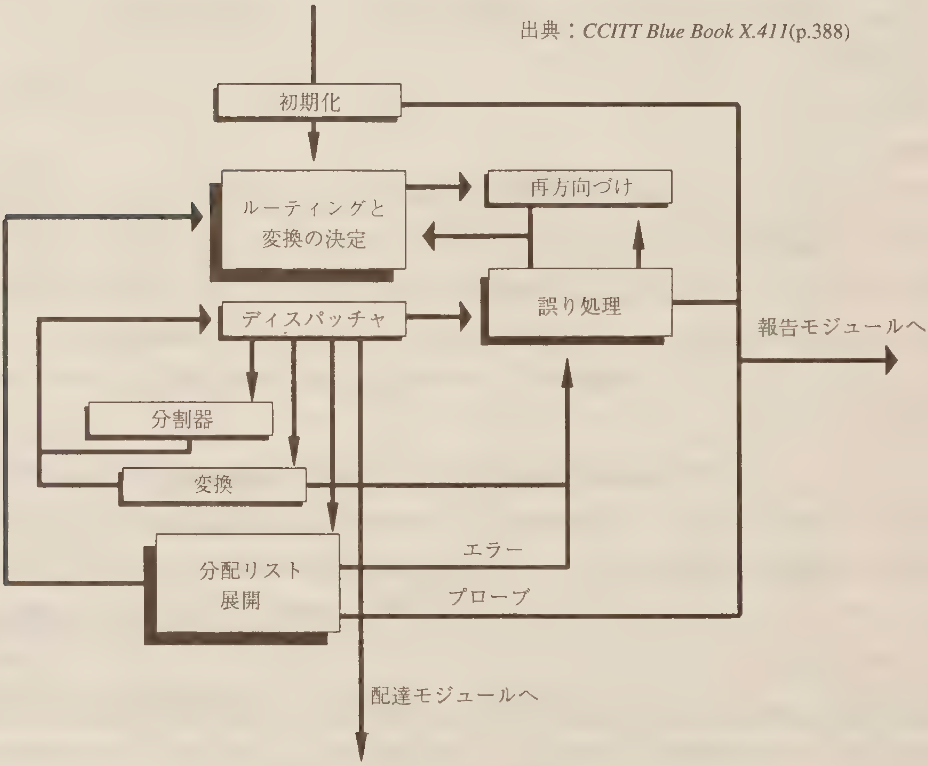
名前の解決，分配リストの展開，再方向づけ，変換などはみな省略可能なサービスである．配達不能は，たいていは宛先ユーザやドメインがわからないことによりメッセージが捨てられるときに，ユーザに対して送り返されるメッセージの例である．

最後に，次にどのメッセージ伝搬エージェント(message transfer agent : MTA)がメッセージを



13-19 MTAの構成要素

出典：*CCITT Blue Book X.411*(p.388)



13-20 メインモジュールの構成要素

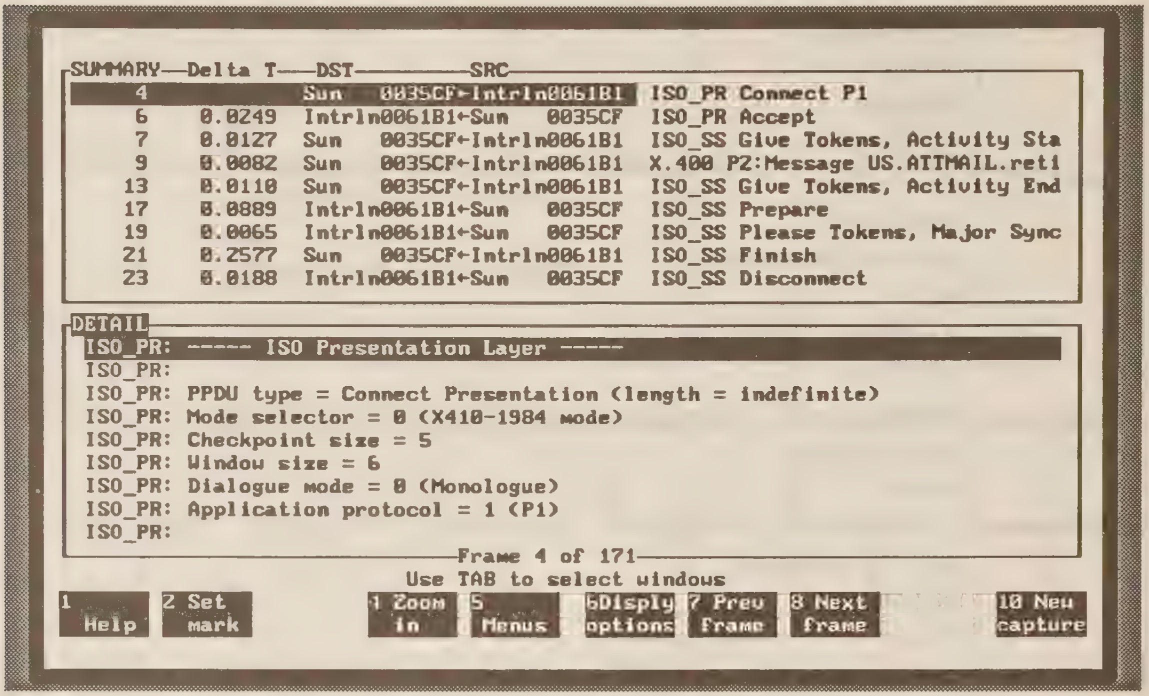
基本項目	アクセス管理
	内容物表示
	変換済表示
	配達タイムスタンプ
	メッセージ表示
	配達不能通知
	本来の符号化情報種別表示
	投函タイムスタンプ
	ユーザ/UAケーパビリティ登録
重要項目	代理受信許可
	遅延配達
	遅延配達の取消し
	配達通知
	他の受信者の公表
	分配リスト展開履歴表示
	配達等級の選択
	プローブ
	要求された配達方法
付加項目	代理受信者が割り当てられている
	内容物機密性
	内容物完全性
	情報が失われる場合の変換禁止
	ディレクトリ名による受信者の指示
	分配リスト展開禁止

13-21　MTAサービス項目

受け取ればいいかという経路選択の問題がある．どのようにして情報を経路づけするかは特定の環境次第である．おそらく，MTA は階層的な経路づけシステムを用いるであろう．すなわち，すべての宛先不明メッセージは，他の管理ドメインに接続されている“ルート”サーバのところへ行く．

図 13-21 に，MTA により提供されるサービスの種類を示す．ここでもまた，サービスは基本項目（常に提供される），重要項目（利用できるが必須ではない），付加項目（実装ごとに省略可能）に分けられる．これらのサービスの項目のほとんどは，MTA のヘッダにより伝達される．例えば，ユーザは代理(alternate)受信者を許す（または許さない）といったことを記述できる．遅延配達(deferrd delivery)は，指定した時刻まで配達しないという但書きをつけてメッセージを MTA に差し出すものである．

遅延配達の取消しはすべての要求が必ずしも満足されるとは限らない項目である．ユーザは遅延配達をつけて，例えば DECnet の Phase V がついに出荷されたという報道発表を送り出すこと



Courtesy of Network General

13-22 X.400トラフィック

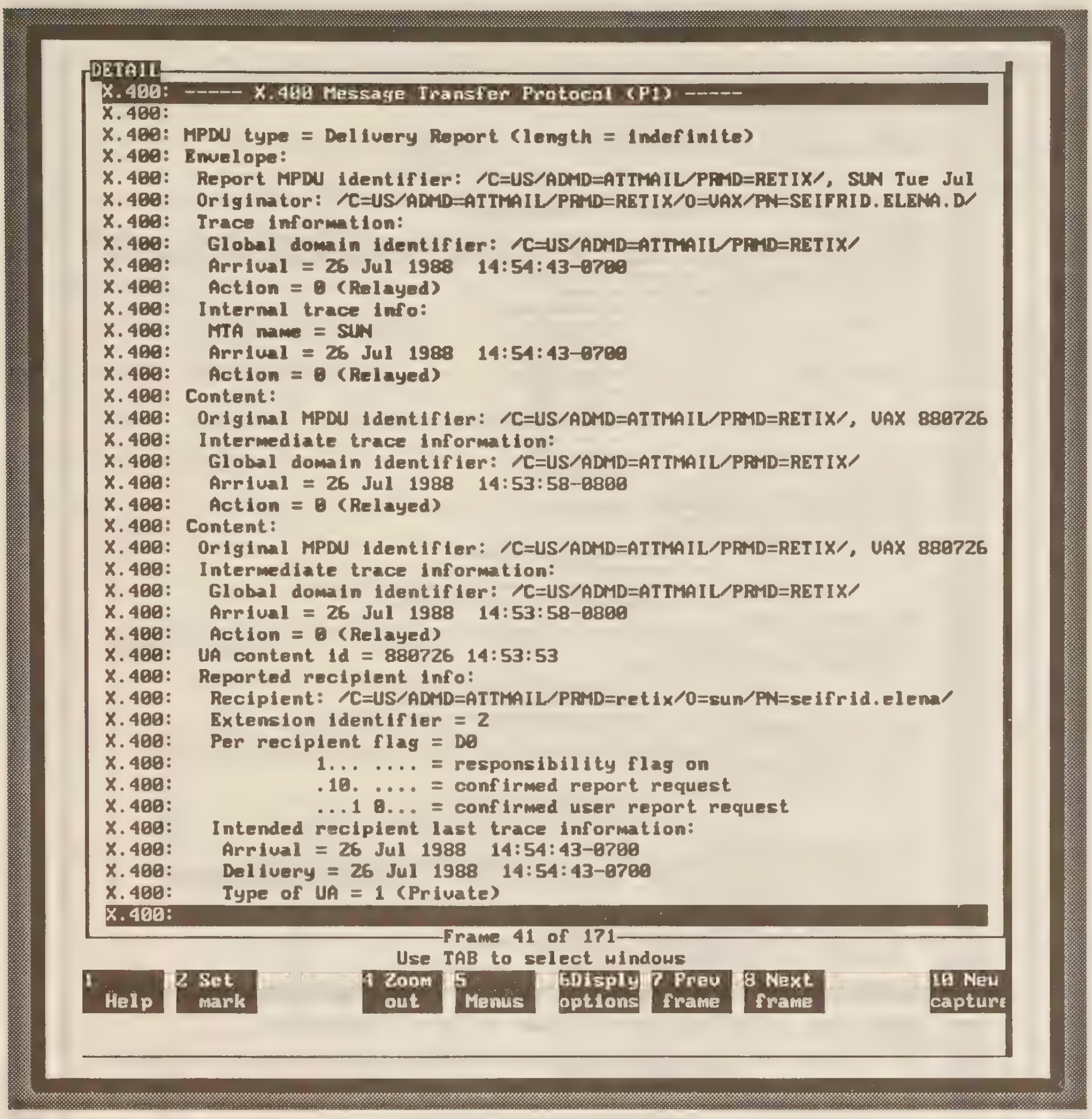
ができる.

ひとたびメッセージが送り出されると，それはメッセージ伝搬システムの中のどこかの待ち行列に入って配達されるのを待つ．このとき送信者が，例えば技術上の問題のために発表を遅らせようとして，遅延配達の取消しを送り出すかもしれない．しかし，遅延配達の取消しが遅延配達メッセージに追いつき，もとのメッセージを取り消すのに間に合うという保証はない．

図 13-22 に，典型的な X.400 のメッセージ交換の様子を示す．最初に，送信側ノードはプレゼンテーションレベルのコネクションを確立し，これが許可される．次に，セション層の活動開始(activity start)が送られる．ここで X.400 P2 プロトコルが呼び出されている．P2 は interpersonal messaging service である．このメッセージは，X.400 プロトコルの 1984 バージョンを用いていることに注意せよ．

図 13-23 は，(MTA 間の)P1 プロトコルをより細かく示している．このメッセージは配達報告である．このメッセージにはメッセージの識別子(メッセージのプロトコルデータ単位の ID)と，レポートの発信者が示されている．また，メッセージがいろいろな場所へ着いたときを示すトレース情報も含まれている．

内容部には，もとのメッセージの識別子が示されている．このメッセージは，United States という country の，ATTMail という administration management domain の，Retix と呼ばれる private management domain から送られたものであり，特定のユーザ ID から送られたものである．



Courtesy of Network General

13-23　X.400 P1プロトコル

メッセージにはまた，Elena Seifrid という personal name をもつ者によりこのメッセージが受け取られ，配達は 1988 年 7 月 26 日になされたということが示されている．

図 13-24 には，メッセージ伝搬プロトコルの中に階層づけられた IPMS が示されている．MTA ヘッダにはメッセージの発信者と基本的な情報種別（ここでは LA5text という基本的なテキストメッセージ）が含まれている．内容種別により，これが独自のプロトコルではなく，IPMS のメッセージであることがわかる．

メッセージにはまた，様々なメッセージ単位のフラグが含まれている．受信者公表(disclose



```
DETAIL
ISO_SS: ----- ISO Session Layer -----
ISO_SS:
ISO_SS: Multi-frame TSDU: frames 9, 11, 12
ISO_SS: SPDU type = 1 (Give Tokens)
ISO_SS: SPDU type = 1 (Data Transfer)
ISO_SS:
X.400: ----- X.400 Message Transfer Protocol (P1) -----
X.400:
X.400: MPDU type = User (length = indefinite)
X.400: Envelope:
X.400:  MPDU identifier: /C=US/ADMD=ATTMAIL/PRMD=RETIX/, VAX 880726 14:53:53
X.400:  Originator: /C=US/ADMD=ATTMAIL/PRMD=RETIX/O=VAX/PN=SEIFRID.ELENA.D/
X.400:  Original encoded information types:
X.400:   Basic information type = A000
X.400:     1... .... .... .... = Undefined
X.400:     .0.. .... .... .... = No tLX
X.400:     ..1. .... .... .... = IA5Text
X.400:     ...0 .... .... .... = No g3Fax
X.400:     .... 0... .... .... = No tIF0
X.400:     .... .0.. .... .... = No tTX
X.400:     .... .0.. .... .... = No tTX
X.400:     .... ..0. .... .... = No videotex
X.400:     .... ...0 .... .... = No voice
X.400:     .... .... 0... .... = No sFD
X.400:     .... .... .0.. .... = No tIF1
X.400:  Content type = 2 (P2)
X.400:  UA content id = 880726 14:53:53
X.400:  Priority = 2 (Urgent)
X.400:  Per message flag = C0
X.400:         1... .... = Disclose recipients
X.400:         .1.. .... = Conversion prohibited
X.400:         ..0. .... = No alternate recipient allowed
X.400:         ...0 .... = No content return request
X.400:  Recipient info:
X.400:   Recipient: /C=US/ADMD=ATTMAIL/PRMD=retix/O=sun/PN=donoghue.barbara/
X.400:   Extension identifier = 1
X.400:   Per recipient flag = A8
X.400:            1... .... = responsibility flag on
X.400:            .01. .... = basic report request
X.400:            ...0 1... = basic user report request
X.400:  Recipient info:
X.400:   Recipient: /C=US/ADMD=ATTMAIL/PRMD=retix/O=sun/PN=seifrid.elena/
X.400:   Extension identifier = 2
X.400:   Per recipient flag = D8
X.400:            1... .... = responsibility flag on
X.400:            .10. .... = confirmed report request
X.400:            ...1 0... = confirmed user report request
X.400:  Trace information:
X.400:   Global domain identifier: /C=US/ADMD=ATTMAIL/PRMD=RETIX/
X.400:   Arrival = 26 Jul 1988  14:53:58-0700
X.400:   Action = 0 (Relayed)
X.400:  Internal trace info:
X.400:   MTA name = VAX
X.400:   Arrival = 26 Jul 1988  14:53:58-0700
X.400:   Action = 0 (Relayed)
X.400:
X.400: ----- X.400 Interpersonal Messaging Protocol (P2) -----
X.400:
X.400: UAPDU type = Interpersonal Message (length = indefinite)
X.400: Heading:
X.400:  IP message id: /C=US/ADMD=ATTMAIL/PRMD=RETIX/O=VAX/PN=SEIFRID.ELENA.
X.400:                 880726 14:53:53
X.400:  Originator: /C=US/ADMD=ATTMAIL/PRMD=retix/O=sun/PN=Seifrid.Elena.D/
X.400:  Primary recipient:
X.400:   O/R Descriptor: /C=US/ADMD=ATTMAIL/PRMD=retix/O=sun/PN=donoghue.bar
X.400:   Reply request = FALSE
X.400:  Primary recipient:
X.400:   O/R Descriptor: /C=US/ADMD=ATTMAIL/PRMD=retix/O=sun/PN=seifrid.elen
X.400:  Subject = Test message on Retix X.400 network
X.400: Body:
X.400:  IA5Text = "Testing for Andre...."
X.400:  Unidentified [1720 bytes]
X.400:
                         Frame 9 of 171
                    Use TAB to select windows
1 Help  2 Set mark  4 Zoom out  5 Menus  6Display options  7 Prev frame  8 Next frame  10 New capture
```

Courtesy of Network General

13-24 X.400 P1およびP2プロトコル

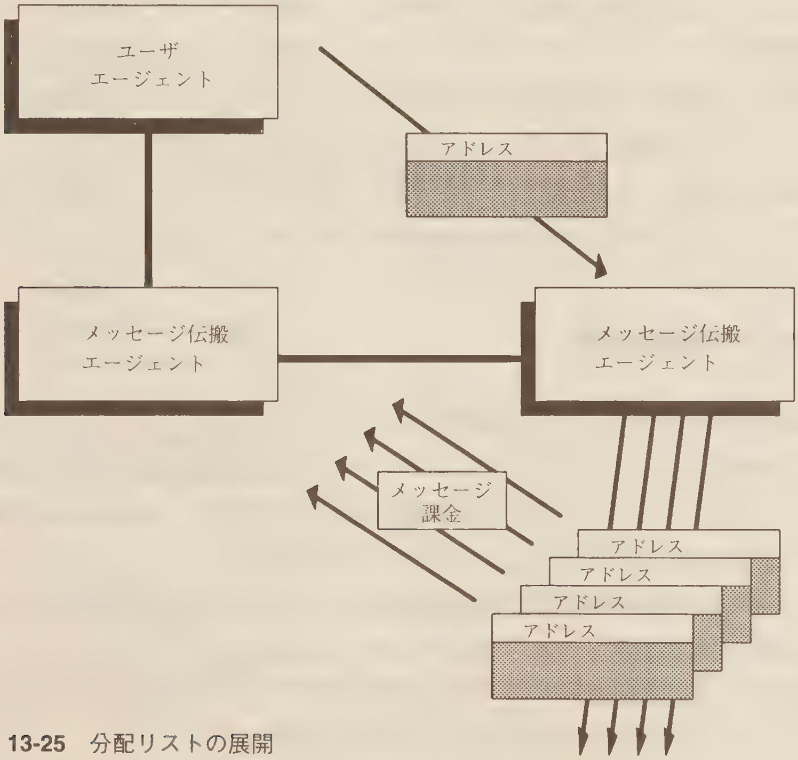
recipients)のフラグがオンになっているので各ユーザが他のユーザについて知ることができることに注意せよ．また，代理受信者は許可されていない．したがってメッセージが配達できなかったときには捨てられる．

次に，2 人のメッセージ受信者の情報があり，そこにはアドレスおよび基本的な報告を要求する(basic reports request)フラグが含まれている．そのあと，メッセージがこれまでどこを通ってきたかを表すトレース情報が続く．このメッセージは VAX と呼ばれる MTA により少なくとも 1 回中継されたようである．

これらすべての情報の後に IPMS のヘッダがある．メッセージには一意の ID，発信者(originator)，および 2 人の主要な受信者(primary recipients)が含まれている．応答要求(reply requested)のフラグは偽に設定されている．もちろんフラグが真に設定されていたとしても，ユーザに強制的に応答させる方法はない．メッセージの本文(body)は，1 つの本体部から構成されていて，LA5Text により作成されている．メッセージは“Testing for Andre”で始まり，それ以外に 1720 バイトある．

13.14 分配リスト

分配リストは，個人の O/R アドレスまたは他の分配リストの一覧表である．メンバーの情報に



13-25 分配リストの展開

加えて，分配リストにはいくつかの特性(properties)がある．

・許可情報(submit permission)には，どのユーザや分配リストがこのリストを利用できるかが記されている．

・展開地点(expansion point)はこのリストのO/Rアドレスである，すなわちどこでこのリストを構成メンバーに変換するかということを示している．

・リストの持ち主(owner)

メールを送ったユーザは，その宛先アドレスが分配リストであるということを知る必要はない．しかし，もし1つのメッセージが100のメッセージになってしまうとしたら，料金は急速に膨れ上がるということを覚えておいてほしい．X.400のオプションの1つに，分配リストの展開を禁止するという項目がある．

分配リストを展開する地点はMTAである(ユーザが存在しないからである)．展開地点ではメッセージをいくつかに“分解”し，これらをメッセージ伝搬システムに送り出す(図13-25参照)．これらの活動に関わる一切の料金は，通常，送信元のユーザにかかる（分配リストの持ち主のせいで，損害を被る可能性があるにもかかわらずである)．

分配リストは入れ子にすることができる．入れ子になった分配リストを展開するときには，メッセージの送信者ではなく親分配リストのIDが，展開を許すかどうかの検査に使われる．分配リストの潜在的な問題は再帰(recursion)である．これを防ぐために，メッセージはこのリストをその地点まで伝えるのに用いられた分配リストのチェーンを含んでいる．展開者はリストに再帰がないかどうかを調べる．

分配リストに関する通知は，分配リストの展開地点(許可されなかった場合など)，または最終的な目的地（メッセージが配達不能であった場合）から送られてくる．

分配リストは特別な形をしたO/Rアドレスである．そのO/Rアドレスに関して責任をもっているMTAのところへ届くまで，メッセージは他のものと同じように扱われる．

13.15　メッセージ格納領域

メッセージ格納領域は，PCのような稼働率の低いワークステーションの場合に有効である．PCはメッセージを常時受け取ることはできないため，PCが1つ仕事を終えメールを扱えるようになるまで，何らかの形でメッセージを蓄えておく方法が必要である．メッセージ格納領域は，コンピュータをもたず端末だけをもっているユーザにも有効である．

図13-26に，メッセージ格納領域により提供されるサービスの項目を示す．見てわかる通り，サービスはかなり簡単なものである．ユーザはメッセージを格納したり，消去したり，取り出したり，一覧表を出したりすることができる．これに加えてメッセージ格納領域から警告を送ったり，自動転送を行ったりする機能もある．

基本項目	格納されているメッセージの消去
	格納されているメッセージの取出し
	格納されているメッセージの一覧
	格納されているメッセージの要約
付加項目	格納されているメッセージの警告 (alert)
	格納されているメッセージの自動転送
出典：CCITT Blue Book VIII.7-X.400 (p.37)	

13-26 メッセージ格納領域の要素

メッセージ格納領域はこのように，メッセージをメッセージ伝搬システムから取り出して，ユーザの代わりにこれを受け取るプログラムということができる．

13.16 ゲートウェイ

ゲートウェイは，異なるメッセージ処理システムの間の相互接続を行う．例えばテレックスのユーザは，テレックスアクセスユニットを介して X.400 のユーザにアクセスすることができる．おそらくゲートウェイの中で最も興味深いのは，郵便配達サービスとの相互接続であろう．

明らかに，この種のゲートウェイは一方通行である．すなわち，どんな形式で書かれているか

基本項目	基本的な物理媒体による表現
	通常のメール
	物理的な転送許可
	物理的なメッセージの返還を伴う配達不能メール
重要項目	カウンタ収集
	速達メールサービス
	特別配達
付加項目	別の物理媒体による表現
	アドバイス付きカウンタコレクション
	Bureaufaxサービスによる配達
	MHSによる物理的配達通知
	PDSによる物理的配達通知
	通知の転送禁止
	書留
	個人宛の書留
	転送アドレスの要求
出典：CCITT Blue Book VIII.7-X.400(p.36)	

13-27 郵便ユニット項目

わからない紙片を自動的にX.400のメールメッセージにすることなどほとんど不可能だからである．しかし，逆はほとんど自明である．ゲートウェイは入ってくるメッセージを取り込み，本体部分を紙に印刷し，郵便アドレスを表につけ，何らかの手段で郵便サービスにこれを渡して配達してもらえばいいだけである．

ユーザにとってこの種のゲートウェイの利点は大きい．つまり1つのユーザインターフェイスだけですべてのメッセージ受信者と交信が行えるのである．

著者はX.400郵便配達システムのアクセスユニットが実装されているのを見たことはないが，MCI Mailは似たようなサービスを提供している．メッセージを送るとき(あるいは分配リストを構成するとき)，MCI Mailのユーザは単に名前を書くだけでよい．もしその名前が有効なMCI Mailアドレスでなければ，システムは配達システムの名前を尋ねてくる．オプションとして有効なものにはX.400，テレックス，ファックス，そして郵便配達がある．

郵便配達の場合，システムはユーザに正しいアドレスを尋ねてくる．メッセージ本文と封筒が印刷され，メッセージは郵便サービスへと送り出される．料金はMCI Mailにより支払われ，ユーザへの毎月の請求書により回収される．

図13-27に，郵便配達ゲートウェイのサービスの基本項目を示す．明らかにアクセスユニットは，メッセージの基本的な物理媒体による表現が行えなければならず，また郵便システムがメールの転送など通常の処理を行えるようにしなければならない．

ユーザはカウンタ収集(counter collection)，速達(express mail service)，さらには特別配達(special delivery)などの特別オプションも利用できるようになっているべきである．アクセスユニットの提供者がこれらのサービスの料金を払い，あとでX.400のユーザに請求するのだということを忘れないでほしい．

書留や転送アドレスの通知といった各種の付加項目については，郵便サービスとの間で多少緊密な協力が必要となる．基本項目は地方サービス局が容易に提供することができる．しかし付加項目は，そのサービスが郵便サービスにより提供されているか，少なくとも認可されていなければならない可能性がある．

13.17 X.400へのmunging

DNS，SMTPおよびRFC822に基づくInternetの世界をX.400に接続するというのはきわめて挑戦的な試みである．Sunの世界では，X.400への接続はSunLinkのMHSソフトウェアというアクセスユニットにより実現されている．これはX.400の世界と本来のSMTPまたはUUCPに基づく環境とを結ぶ単なるゲートウェイである．

どうやってこれら2つの世界をゲートウェイを介して接続するかに関しては，非常に複雑な2つのRFCが議論している．Internetの世界からX.400の世界へと進むことは，それほどひどく複

雑ではない．これにはゲートウェイをまたぐアドレスの変換を処理する方法が必要となる．このことを除いては，RFC822 の単純なヘッダ項目のほとんどはきわめて容易に X.400 の世界へとマップすることができる．

難しいのは X.400 から RFC822 への変換である．X.400 の方がより多くの機能を提供している．とりわけ問題となるのは，X.400 では本体部分に単なる ASCII 以外のものも許しているということと，1 つのメッセージがいくつかの本体部分から構成され得ることである．

一般的な規則として，テキストの変換は非常に簡単である．本体部分が変換できなかった場合，ゲートウェイはメッセージを拒否するか，できる範囲で変換して，誤りが起こった場所を示すために，“ゲートウェイがこの部分を喰べた”標識を挿入する．

```
From ICS.UCI.EDU!na-mhsnews-request  Wed Oct 24 15:43:24 1990 re-
mote from uunet
Received: by malamud.com (UUL1.3#5653)
   from uunet with UUCP; Wed, 24 Oct 90 14:37:58 PST
Received: from ics.uci.edu by uunet.uu.net (5.61/1.14) with SMTP
   id AA18872; Wed, 24 Oct 90 15:43:24 -0400
Received: from ics.uci.edu by ICS.UCI.EDU id aa08180; 24 Oct 90 10:21
PDT
Received: from ics.uci.edu by ICS.UCI.EDU id aa08167; 24 Oct 90 10:19
PDT
Received: from kwai.inria.fr by ICS.UCI.EDU id aa08102; 24 Oct 90 10:19
PDT
X400-Received: by /PRMD=inria/ADMD=atlas/C=FR/;
   Relayed; 24 Oct 90 18:20:38+0100
X400-Received: by /PRMD=emse/ADMD=atlas/C=fr/;
   Relayed; 24 Oct 90 18:14:48+0100
Date: 24 Oct 90 18:14:48+0100
From: Paul-Andre Pays <uunet!mars.emse.fr!pays>
Message-Id: <9010241714.AA00412@mars.emse.fr.emse.fr>
To: mhsnews@ICS.UCI.EDU, mws@sitman.ssw.COM
Subject: re: your mail

—— Begin Included Message ——

X400-Received: by /PRMD=emse/ADMD=atlas/C=fr/;
   Relayed; 24 Oct 90 18:02:05+0100
Date: 24 Oct 90 18:02:05+0100
From: Mark Sitler <mws@sitman.ssw.com>
Message-ID: <9010241502.AA15753@sitman.ssw.com>
To: mhsnews@ics.uci.edu
Autoforwarded: True

Status: RO
```

13-28 3つのMHSを横断するメッセージ

他の項目の変換もまた必要である.例えば,時間は2つのシステムの間で異なる表現の仕方をされている. X.400では, ASCII表現へ変換されなければならない整数に関する規定がある. これらの細かい事柄についても変換する必要がある.

もう1つ難しいのはヘッダの項目の問題である. X.400の世界では, 受信証が送られたり, 変換が実行されたり, 自動転送が行われたりというように, 多くのヘッダの項目により仕事が行われる. これらはRFC822の世界にはほとんど存在しない.

図13-28に, メッセージがX.400からSMTPの世界, そして最後にはUUCPの世界へと渡っていく様子の実例を示す. 見ての通りメッセージ環境の間の相互接続は間違いなく現実のものである. プロトコルがどのように機能しているのかということに関するより正確な情報はMarshall Roseの"*The Little Black Book*"を参照してほしい.

より進んだ勉強のために

David H. Crocker, *Standard for the Format of ARPA Internet Text Messages,* RFC 822, August, 1982.

Frey, Donalyn, and Adams, Rick, *!%@:: A Directory of Electronic Mail* (2nd ed.). O'Reilly & Associates (Sebastopol, CA: 1990).

S. Kille, *Mapping Between X.400(1988)/ISO 10021 and RFC 822*, RFC 1148, March, 1990.

———, *Mapping between X.400 and RFC 822*, RFC 987, June, 1986.

LaQuey, Tracey L., ed., *The User's Directory of Computer Networks.* Digital Press (Maynard, Mass: 1990).

Tim O'Reilly and Grace Todino, *Managing UUCP and Usenet,* O'Reilly and Associates (Newton, Mass: 1988).

Craig Partridge, *Mail Routing and the Domain System,* RFC 974, January, 1986.

J. Postel, *Simple Mail Transfer Protocol,* RFC 821, August, 1982.

Quarterman, John S., *The Matrix.* Digital Press (Maynard, Mass: 1990).

Rose, Marshall, *The Little Black Book: Mail Bonding with OSI Directory Services.* Englewood Cliffs, N.J.: Prentice Hall, 1991.

Marshall T. Rose, Einar A. Stefferud, *Proposed Standard for Message Encapsulation,* RFC 934, January 1985.

Sun Microsystems, *SunNet MHS System Administration Manual,* Part No: 800-5009-1300, Version 50 of 05 October 1990.

Grace Todino, *Using UUCP and Usenet,* O'Reilly and Associates (Newton, Mass: 1987).

Chapter 14

ファイル/プリンタ/端末

14.1 概要

この章も12章と同様に寄せ集めである．ここではInternet上でよく使われるアプリケーションについて説明する．それには2つの基本的なサービスである仮想端末サービス(Telnet)とファイル転送プロトコル(File Transfer Protocol : FTP)が含まれている．InternetのアプリケーションであるFTPに加え，OSIで規定されているファイル転送，アクセスおよび管理(File Transfer Access and Management : FTAM)プロトコルも付記されている．Sunのコンピュータは徐々に，TCP/OSIの両環境で使われるようになってきている．SunLink OSIゲートウェイのソフトウェアは，FTAMや他のOSIベースのサービスへのアクセスにも対応するようになっている．

まずTelnetから始め，その後FTPやFTAMの説明を行う．その2つのファイル転送プロトコルの後で，寄せ集めの中のさらなる寄せ集めまで解説する．まずリソースの位置に関する2つのプロトコルであるwhoisとfingerについて説明する．次にリモートプリントをする場合のラインプリンタデーモン(LPD)プロトコルについて，そして最後にディスクレスノードをブートするのに使う自明ファイル転送プロトコル(trivial file transfer protocol : TFTP)や他の様々なタスクについても議論を行う．

14.2 Telnet

TelnetはTCPベースのサービスで，ユーザはところどころに制御情報が入ったデータを送ることができる．典型的な使用例はリモートログインだが，そのサービスは一般的なものであるので，様々なアプリケーションに使うことができる．Telnetには次の3つの基本的考え方がある．

- ・ネットワーク仮想端末(Network Virtual Terminal：NVT)
- ・ネゴシエートオプション
- ・端末の対称性

ネットワーク仮想端末はクライアントとサーバの両方が操作できるような抽象的デバイスを定義している．ユーザが入力をすると，その情報はTelnetのクライアントプロセスによってネットワーク仮想端末上のオペレーションに変換される．どうしてそのような抽象化をするのだろうか？　Unixの世界では異なる端末間では大変多くの違いがあるため，端末を管理することは非常にデバイスに依存した操作になる．共通の仮想的オペレーションセットを使うことにすれば，それぞれの実装者はデバイス依存のそれぞれのオペレーションを，取り決めたセットに置き換えることができる．

ネゴシエートオプションが意味するところは，Telnetプロトコルが開始されるときには非常に単純だが，徐々に複雑になっていくということである．クライアントとサーバが両方で同意したならば，ベルやホイッスル(のコード)が特殊制御キャラクタとして加えられ，より高機能なネットワーク仮想端末になる．同意しなければ，ネットワーク仮想端末は単純な端末のままになる．

対称性というのは，接続されているどちらの側も使用するオプションを提示できることである．このことで容易にループにはまる危険性があるので，ルールがいくつか決められている．例えば，Telnetプロセスは何かを変更する場合にのみオプション要求を送信でき，そのときのモードを通知するだけの目的には送信できないとすることである(さもないと通知が通知を引き起こし，ループを生じさせてしまう)．

ループを防ぐ別の方法は，ノードがすでに入ってしまっているモードに入れという要求を受け

コマンド	コード	意味
SE	240	サブネゴシエーションパラメータの終了
nop	241	オペレーションなし
data mark	242	syncのデータストリームの部分(TCP緊急ポインタも設定すること)
break	243	Network virtual terminalのブレークポインタ
ip	244	割込みプロセス
ao	245	出力のアボート
ayt	246	Are you there ?
ec	247	文字削除
el	248	行削除
ga	249	Go Ahead
sb	250	サブネゴシエーションの開始
will	251	ネゴシエーションのコマンドで，オプションコードが続く
wont	252	
do	253	
dont	254	
iac	255	コマンドとして解釈

14-1 Telnetコマンドキャラクタ

取った場合には，そのノードは応じないことである．応じなければ，要求の無限ループを防げる．モードの変更の要求に対しては，ただその変更を拒否する場合でも，必ず返事を返さなければならない．これらのルールに従えば，ループを防げる他にもオプションの嵐を防いだり，少なくともそれをセッションの始めだけに限り，後は徐々におさまるようにできる．

オプションは次の4つの動詞がベースになっている．

- will：このサービスを開始したい．
- do：このサービスを開始しなさい．
- won't：このサービスは開始したくない．
- don't：このサービスの開始はするな．

Telnetプロセスはwillオプションを送って，何かをしたいことを知らせる．その返事がdoオプションだったときは，ネゴシエートは成功したことになる．返事がdon'tオプションだったときは，ネゴシエートは不成功である．同様に，リモートノードに何かをさせたい場合は，そのノードはdoオプションを送り，その返事はwillオプションかwon'tオプションのどちらかになる．

この構造を使えば，ノードは知らないオプションは拒否することができる．いったんその取り交わしが行われた後で，そのサブオプションのネゴシエーションが行われることがあるが，それはそのようなサービスを行うことを両者が同意した場合だけである．

14.2.1 ネットワーク仮想端末

ネットワーク仮想端末(Network Virtual Terminal : NVT)は出力装置やキーボードの備わった双方向キャラクタデバイスである．出力装置は入力されてくるデータに応答し，キーボードは送出データを作り出す．送出データは，エコーが設定されている場合には出力装置にも送られることに注意してほしい．下層のトランスポートサービスは本来は全二重であるが，デフォルトのNVTでは半二重のデバイスオペレーションで，ラインバッファモードが使われている(ローカルに行ごとにため，それを一度に送り出す)．オプションによってこれをキャラクタバッファデバイス(フルスクリーンエディットのため)や全二重デバイスに変更することができる．

基本的データストリームは7bitのASCIIキャラクタで，ASCIIキャラクタセットの下半分に範囲が限られている．さらに，ところどころデータの中に連続した制御文字も入っている(図14-1参照)．すべてのコマンドは特殊なInterpret As Command(IAC)制御文字で始まり，1バイト以上の制御情報が後に続く．

Telnetで提供される特殊シグナルはsyncシグナルである．syncはTCPの緊急通知で，Telnetデータマーク(DM)と組み合わせられる．緊急通知はTCPのフローコントロール下には置かれない．リモートノードがそれを受け取ると，やってきているデータストリームの中から注意すべきデータ，例えばアボート制御信号をすぐに探し始める．

データマークは対象領域の終わりを知らせる．普通は，これはリモートノードに終わりを知らせ，通常の処理を再開させることになる．例えば，ユーザがコマンドをリモートのオペレーティングシステムにあるコマンドを送るとする．そのコマンドが48時間かかるかもしれないというの

がわかったとき，ユーザはローカルのアボート制御信号，つまりコントロールCをたたく．

Telnet モジュールはコントロールCを受け取り，それは特殊キャラクタであることを理解する．そのキャラクタを発送してリモートホストが最初のコマンドを終える(そして最後にはそのアボートシグナルを受け取ることになるが)まで単に48時間待つよりも，Telnet は TCP 緊急通知を発送するだろう．リモートの Telnet はそれを理解して，リモートプログラム(この場合はコマンドプロセッサ)に割込みがかかったことを知らせる．それはデータマーク(コントロールC)までの残りのすべてのデータをリモートのオペレーティングシステムに渡すが，このデータマークのためにおそらくはそのコマンドをアボートするだろう．データマークは対象とするデータの終わりを示しており，ユーザは次のコマンドを続けることになるだろう．

14.2.2 サブネゴシエーションとオプション

様々なRFC(図14-2参照)の中で様々の異なるオプションが定義されている．標準の Telnet

ドキュメント	タイトル
RFC 1116	Telnet ラインモードオプション
RFC 1097	Telnet サブリミナルメッセージオプション
RFC 1096	Telnet Xディスプレイ位置オプション
RFC 1091	Telnet 端末タイプオプション
RFC 1080	Telnet リモートフローコントロールオプション
RFC 1079	Telnet 端末スピードオプション
RFC 1073	Telnet ウィンドウサイズオプション
RFC 1053	Telnet X.3 PAD オプション
RFC 1043	Telnet データエントリ端末オプション:DODIIS インプリメンテーション
RFC 1041	Telnet 3270 レジメオプション
RFC 946	Telnet 端末ロケーションナンバーオプション
RFC 933	マーキング出力 Telnet オプション
RFC 930	Telnet 端末タイプオプション
RFC 927	TACACS ユーザ定義 Telnet オプション
RFC 885	Telnet レコード終了オプション
RFC 884	Telnet 端末タイプオプション
RFC 861	Telnet 拡張オプションリストオプション
RFC 860	Telnet タイミングマークオプション
RFC 859	Telnet ステータスオプション
RFC 858	Telnet Go Ahead 抑制オプション
RFC 857	Telnet エコーオプション
RFC 856	Telnet バイナリ転送
RFC 855	Telnet オプション指定
RFC 854	Telnet プロトコル指定

14-2 Telnet オプション

RFCのものもあるが，その他多くは端末セッションに関するハードウェアやソフトウェアが増えるにつれて後から定義されたものである．

サブネゴシエーションプロトコルはそれらのいくつかのために定義されているものである．どちらの側もそのオプションの実行に同意し，それからサブネゴシエーションに自動的に入る．サブネゴシエーションはSB(250)とSE(240)バイトを使って取り交わされる(それぞれ直前にIACを伴う)．サブネゴシエーションの1つの例はオプションのリストを拡張することである．定義されているオプションは254あるが，extended options list(EXOPL)コードを使えば，さらに256オプションを追加することができる．

IAC SB EXOPL DO option IAC SE

このコマンドはIACから始まり，コマンドが続くことを示している．次に，subnegotiation begin (SB)は特定のサブネゴシエーション，ここではEXOPLが続くことを示している．DOコマンドはそのノードがそのオプションを実行したいことを表す．最後にはIAC subnegotiation end(SE)がきて普通のデータ処理を再開できることを示している．

その拡張オプションが順にネゴシエーションを続ける必要がある場合は，次のようなものが交わされるだろう．

IAC SB EXOPL SB OPTION PARAM SE IAC SE

再び，サブネゴシエーションが知らされ，EXOPLが入っている．それに，オプションとパラメータの付いたネストされたサブネゴシエーションが続く．最初のサブネゴシエーションが終わると，外側のサブネゴシエーションが続く．見た目には少々混乱するが，プログラムの関係を説明するのに簡単である．

デフォルトでは，SunのTelnetの実装ではリモートエコー，先打ちの抑制，そして端末のタイプなどのオプションを要求してくる．さらにエコー，バイナリ，先打ちの抑制，そしてタイミングなどのオプションも扱える．

14.2.3 出力の破棄

多くのオペレーティングシステムはコントロールOのようなものを備えている．これはそれより先の出力をすべて捨てることである．Telnetプロトコルで接続された端末の前で，ローカルに直接接続されたシステムと同様にリモートシステムでも同じ振舞いをしてくれることを期待するユーザは多いだろう．

問題は，アボート出力キャラクタがホストのオペレーティングシステムに到着するまでには，Telnetサーバによってすでに何千ものキャラクタが送られており，それがユーザの端末には表示されないことである．クライアントシステムはリモートホスト上で出力を捨てるのにどのキャラクタを使うかを知らないので，普通のデータとしてコントロールOを流すだけになってしまう．

output discardオプションは特殊なoutput discard信号が入力されたことをクライアントシス

テムが知る1つの方法である．Telnet クライアントは，リモートオペレーティングシステムのどの特殊コマンドキャラクタが出力アボートや割込みプロセスの抽象的ファンクションに割り当てられているかを知らなくてはならない．

ここでの潜在的問題は，Telnet データストリームにはオプションネゴシエーションも含めて，コマンドが埋まってしまっていることである．これは，ノードは全データを無差別に捨てるわけにはいかないことを示している．“普通”のデータだけが捨てられるようにすべきである．これを扱う1つの方法は，リモートホストに discard output キャラクタを見つけさせることである．ホストオペレーティングシステムが discard 信号を発するとすぐに，Telnet サーバは Telnet クライアントに discard が行われていることを速やかに知らせなければならない．その通知はクライアントに送られる直後の TCP セグメントに入れられる(サーバに TCP データが多量にバッファされていてもである)．

この問題を扱うよりよい方法は，TCP パケットで緊急ビットを使うことである．TCP 緊急ビットはデータストリームの中で，対象とするバイトを区別する．その対象としているバイトには2キャラクタシーケンスである IAC DM が入っている．IAC は制御文字がデータストリームの中に現れることを示す．データマーク(DM)は Telnet が挿入したマークで，同期信号として働く．ホストは DM までの情報はすべて捨ててよいことがわかる．

Telnet クライアントシステムは，オペレーティングシステムに固有なファンクションに割り当てられたコマンドキャラクタが来たとわかると，このオペレーティングシステム固有のキャラクタを標準の Telnet の表現に変換する．

- IAC AO(abort output)
- IAC IP(interrupt process)

クライアントはオペレーティングシステム固有のキャラクタを見つけると，適当な表現を挿入し，データストリームにそれを流し，その直後に sync シグナルを付け加える．Telnet サーバはその abort output を受け取ると，クライアントに sync を送り返す．IP を受け取った場合，sync 信号を送り出すこともある．

14.2.4 ワークステーションのサポート

Telnet のいくつかのオプションを使えば，ワークステーションのウィンドウシステムの中から Telnet を利用できる．これらのオプションで，ディスプレイのウィンドウサイズやサイズの変更についての情報を，クライアントからサーバへ送ることができる．RFC 1075 で指定される Negotiate About Window Size(NAWS)では，ウィンドウの大きさは 65536×65536 が許される．

他のオプションの RFC 1091 を使えば，既存の Telnet Terminal Type(TTTYPE)を拡張し，クライアントの端末エミュレーションのサポートリストをサーバが参照して好きなものを選択できる．後で(つまりアプリケーションが変更を要求したとき)，サーバは新しいタイプを選ぶことが

できる.

RFC 1096を使うと，ユーザがあらためて操作しなくとも，Telnetのコネクションを通じてX-ウィンドウシステムのディスプレイの所在を送ることができる.新しいオプションであるX-Display-Location(XDISPLOC)は，Telnetのクライアントとサーバがネゴシエートできるような仕様になっている．クライアントとサーバがそのオプションのネゴシエートに同意すると，サーバはTelnetサブネゴシエーションシーケンスを送出して

host:display_number[.screen_number]

という形式のASCII文字列の問合せを行う.

14.2.5 Telnetトラフィック

図14-3から14-6までは典型的なTelnetのトラフィックを示している．図14-3は2つのノードの間のトラフィックのいくつかのフレームを表している.トラフィックは1回に1文字, 1文字に1パケットずつ，エコーバックされていることに注意してほしい．図14-4はTCP上のTelnetセッションを示す．パケット確認応答に注意すること.

図14-5と14-6はオプションのネゴシエーションを示している.最初のノードは,トラフィックをエコーしたり，端末のタイプを設定するなどのテーブル上の項目をいくつか指定している．図14-6では，Send Locationオプションが拒否されたことを表す.

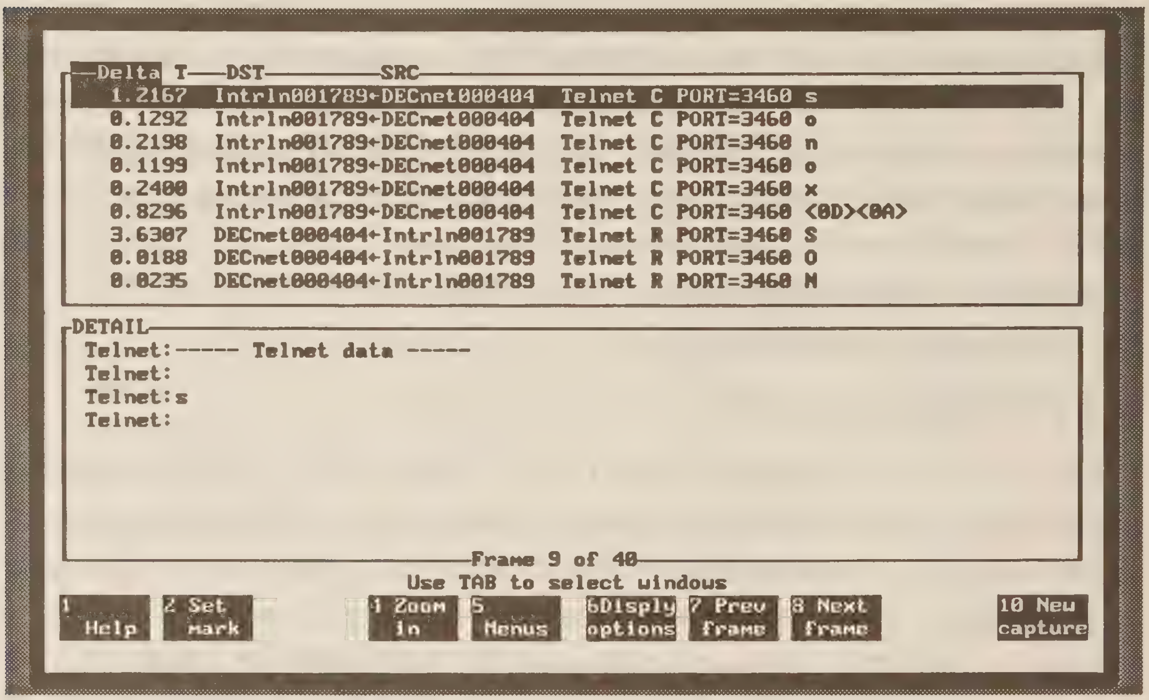
14.3 ファイル転送プロトコル

ファイル転送プロトコル(File Transfer Protocol : FTP)はMailやTelnetと同様，TCP/IPの3大アプリケーションの1つである．FTPは1971年のRFC 114まで遡る．時間とともに細部の変更はあったものの，FTPはInternetで大きな役割を担っている．他にも特にNetwork File SystemやFTAMなど，多くのデータアクセス機構がある．Network File Systemはアクセス機構のより統合化された種類で，第6章でもっと詳しく扱っている.

FTAMはFTPの後で議論するが，FTPとは異なるサービスを提供する．FTAMサービスはOSI上でファイルアクセスを扱う.それはレコード, 属性, そしてその他のファイルの構成要素である．FTPは対照的により限られた範囲で，全ファイルの転送を対象とする.

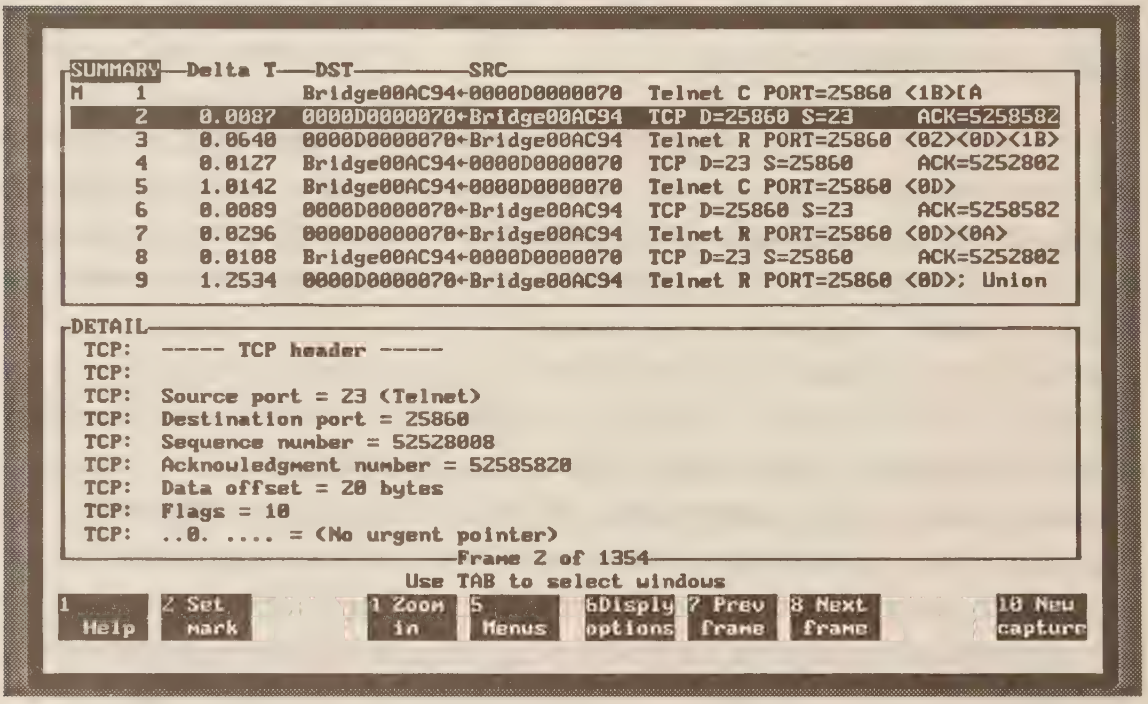
FTPはFTAMに比べてより重要な利点をもつ．すなわちそれは，より広く実装されている. TCP/IPパッケージの標準的要素として, 主要なオペレーティングシステム上のすべてで, そしてパブリックドメインバージョンでもしばしば手に入れることができる．仕事場や企業のネットワークでは, FTPの多くはNFSにとって換わられてしまったが, セキュリティや管理境界のためにNFSのアクセスできないようなネットワーク間の場合などいまだに広く使われている.

FTPは2つのTCPコネクションをベースとしている. 1つはTelnetコネクションで，コマンド



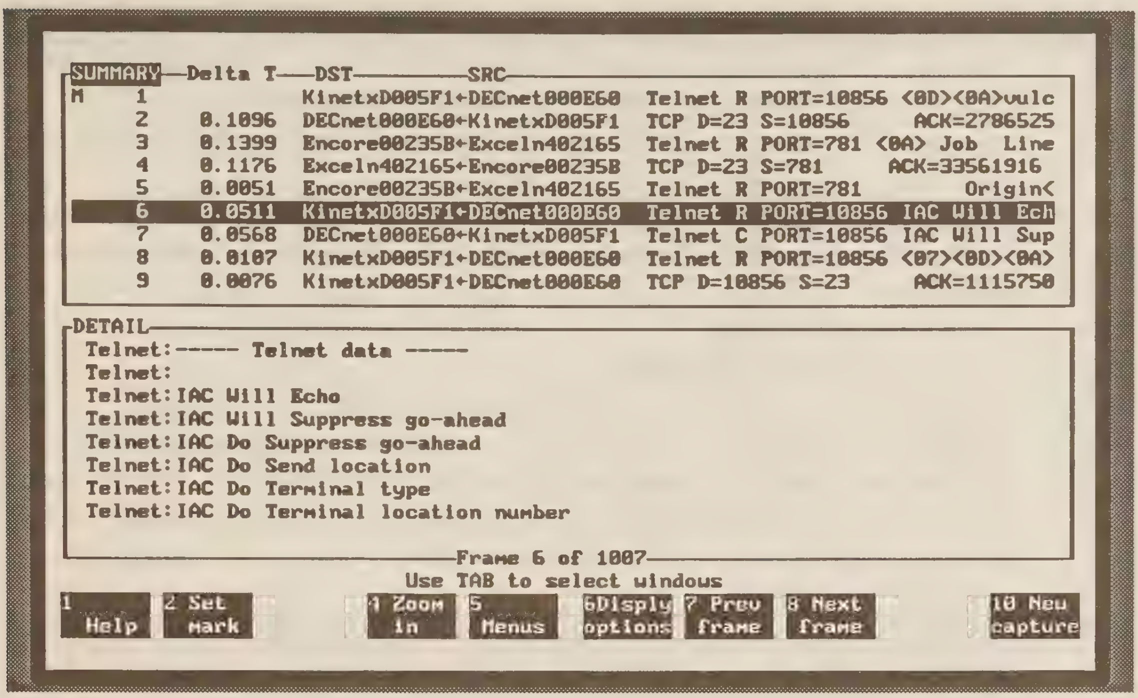
Courtesy of Network General

14-3　遠隔エコー



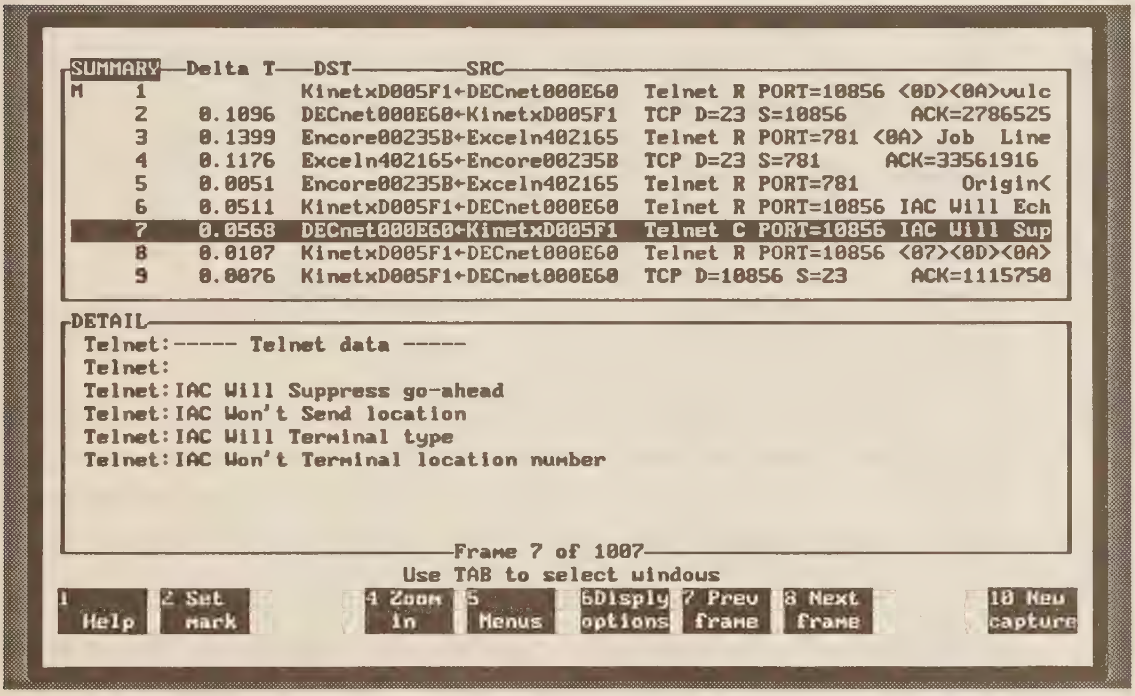
Courtesy of Network General

14-4　Telnet over TCP



Courtesy of Network General

14-5 Telnetオプションのネゴシエーション



Courtesy of Network General

14-6 ネゴシエーションに対する応答

アクセスコントロールコマンド	
User	システムによって要求される．ユーザ名だけでよいが，パスワードやアカウント情報が要求されることがある．サーバによってはどの時点でも新しく user コマンドを入力することができる．
Pass	Password．直前に username コマンドがなくてはならない．user FTP では user 画面からマスクされるべきである．
ACCT	アカウント情報．これを自動化する方法がある．PASS コマンドへの返答が 332 だった場合にはアカウント情報が要求される．応答が 230 だった場合には要求されない．後で要求されると，332 か 532 応答が返ってくる．
CWD	ワーキングディレクトリの変更．
CDUP	上のディレクトリに移る．CWD の特殊なケース．
SMNT	ストラクチャマウント．異なるファイルシステムデータストラクチャをマウントする．
REIN	再初期化．ユーザを中断させ，すべての I/O とアカウント情報を廃棄する(転送は最後まで続けられる)．すべてのパラメータはデフォルトの設定になり，コントロールコネクションはオープンしたまま残される．
QUIT	ログアウト．ファイル転送中でなければ，コントロールコネクションをクローズする．転送中だったら，結果の応答を受け取るためにオープンされたままになる．相手のユーザとセッションを続けたい場合は，quit の代わりに REIN を使用．
転送パラメータコマンド	
PORT	データポート．デフォルトはユーザのソースポート．使用する場合は，32ビットの IP アドレスと 16ビット TCP ポートナンバーから構成する．これは 8ビットフィールドに分解され，コンマで区切られた 10 進数で送られる．
PASV	パッシブ．デフォルトでないデータポートを聞き，転送コマンドの応答に従い伝える代わりに，コネクションを待つ．
TYPE	表現タイプ．2番目のパラメータをもつものもある．最初のパラメータは1文字で，2番目は1文字か数字(デフォルトでASCII nonprint)．
	A(ASCII)
	C(ASA キャリッジコントロール)
	E(EBCDIC)
	I(Image)
	N(nonprint)
	T(Telnet フォーマットエフェクタ)
L	ローカルのバイトサイズ．
STRU	ファイルストラクチャ(F がデフォルト)
	F ファイル．
	R レコード．
	P ページ．
MODE	転送モード(S がデフォルト)．
	S ストリーム．
	B ブロック．
	C コンプレス．

14-7　FTP コマンド

FTP サービスコマンド(引数は普通はパス名)	
RETR	引き出し.
STOR	格納. サーバ DTP にデータを受けさせる.
STOU	ストアユニーク. コピーは作るが, ファイル名はユニークにする.
APPE	アペンド(作成して). サーバはデータを受け取り, サーバ側に格納する.
ALLO	アロケート. 十分な記憶場所をユーザに割り当てる. 引数は確保する論理バイト数の 10 進法表現. ファイルがレコードやページストラクチャの場合は, 2番目の引数は最大のレコード／ページサイズの論理バイト数. store や append コマンドを伴う. オペレーティングシステムがファイルのために領域を確保する必要がない場合は, NOOP とみなされる.
REST	リスタート. 引数はサーバマーカ.
RNFR	～からリネーム. 直後に RNTO コマンドを伴う.
RNTO	～へリネーム.
ABOR	アボート. カレントの FTP サービスコマンドがすでに終了している場合, 応答は 226 (アボート処理正常終了)になる. 転送中は, 応答は 426 (サービスリクエスト異常終了)で, 226 が続く.
DELE	削除.
RMD	ディレクトリの削除.
MKD	ディレクトリの作成.
PWD	ワーキングディレクトリの表示.
LIST	ディレクトリ中のファイルのリスト(あるいは, 引数にファイル名が入力された場合にはそのファイルの情報)を表示する. 引数がない場合はユーザのカレントワーキングやデフォルトディレクトリ. データ転送はデータを通じて行われる.
NLST	ネームリスト. ファイルネームを CRLF や NL で区切ってストリームにして返す. プログラム中で使用(すなわち multiple get ファンクション).
SITE	サイトパラメータ. サイトの指定. help site コマンドで見つけること.
SYST	システム. オペレーティングシステムの名前を返す.
STAT	ステータス. status コマンドを送る(ファイル転送中のときもある). 転送中は, 転送中であることを返答する. 転送中でなければ, 引数はファイル名で, ファイルのステータス情報がコントロールを通じて返される.
HELP	パラメータを受け取れる.
NOOP	何もしない.

14-7（続き）　FTP コマンド

を送ったり, 返答をしたりする(図 14-7 を参照). 2 つ目のコネクションは実際にデータをやりとりするのに使われる.

アーキテクチャ自体は 2 つの要素からなっている. それはプロトコルインタプリタとデータ転送プロセスである. プロトコルインタプリタ(protocol interpreter：PI)はコマンド交換に使用される Telnet 通信プロセスである. それぞれのコマンドでどのようなアクションがなされるかを定義して Telnet に価値を付加する.

データ転送プロセス(data transfer process：DTP)はマシン間のデータ移動を扱う. DTP は実際

にはリモートの DTP モジュールを対象にしているのではない．それはファイルの抽象的概念である，ネットワークの仮想ファイルシステムを操作する．この意味するところは，データはマシンに依存しないフォーマットで表現されているということである．

プロトコルインタプリタとデータチャネルをわざわざ分けた利点の 1 つは，ユーザが 2 つの異なるホスト間でファイルを転送できることである．異なるサーバ間でコントロールコネクションがセットアップされ，それから 2 つのサーバ間でデータ接続コネクションをセットアップするためにコマンドが発行される．コントロールコネクションを開いたままにしておけば，引続きモニターをしたり，続くコマンドを発行したりできる．

14.3.1　データ表現

異なるマシンは異なるアーキテクチャをもっているので，FTP はデータ非依存にするための基本的規約を設けている．例えば，バイトサイズは常に 8bit である．システムはバイトをどのように記憶していてもよいが，ネットワーク上では 8bit 表現を用いる．

ASCII(あるいは EBCDIC)コードはテキストの表現に使われる．ASCII コードといえどもまた，システムによって異なっている．そこで ASCII のサブセットで netASCII として知られているものが使われる．これは 7bit の ASCII(印刷可能なキャラクタといくつかのコントロールキャラクタ)で，最上位ビットは常に 0 である．

FTP はユーザが表現のタイプを指定するようにしてデータ表現を扱う．タイプは内部表現(ASCII または EBCDIC)や論理バイトサイズとして知られるバイトサイズを明示的に定義したりできる．論理バイトサイズは転送バイトサイズとは異なる．転送バイトサイズは常に 8bit である．すなわち，データはそれぞれ 8bit で 1 グループになっている．しかしながら，これらの転送バイトのいくつかは大きな論理バイトサイズを構成するのに使われることになるかもしれない．論理バイトサイズが 14(こういう変なアーキテクチャがあったら)の場合は，1 つの論理バイトを表現するのに 2 つの転送バイトを必要とし，2bit は無駄になる．

ASCII と EBCDIC を指定するとき，キャラクタのストリームのフォーマットをコントロールする情報を付加する別のパラメータを，これらのタイプにもたせることができる．

フォーマットコントロールの主な 2 つの種類は，Telnet スタイルと Fortran キャリッジコントロールである(ASA として知られる)．Telnet スタイルはキャリッジリターン，行送り，改行，垂直タブ，そしてフォームフィードのように垂直移動キャラクタを受け付ける．データが画面に直接送られているときや，システムが特殊キャラクタを拾って内部表現に変換する必要があるときには，この種別をすれば受付けが容易になる．ASA フォーマットコントロールでは，行の最初にその行に対する垂直フォーマットコントロールの方法が示されている．これは古いラインプリンタにしばしば用いられたものである．

行の最初の列に来るのが許される基本的なキャラクタは次のように 4 つある．

・空白は紙を1行上げることを意味する.

・0は2行紙を上げることを意味する.

・1は紙の一番上に位置を合わせることを意味する.

・＋は移動なし(重ね書き)を意味する.

FTPでは転送のためのデータ構造を使用できる. 3つの基本的なデータ構造が定義されている.

・ファイル構造(内部構造はない. 連続したデータバイト)

・レコード構造

・ページ構造

ファイル構造がデフォルトで, データは構造なしの連続したバイトからなる単なる文字列であることを意味する. structureコマンドが明示的に使われていないときは, これがデフォルトとなる. レコード構造はデータはレコード列で, それぞれのレコードはキャリッジリターンと行送りの組合せなどのレコード標識で終わっている.

ページ構造はファイルをページに分け, それぞれのページにはヘッダを付ける. ヘッダにはそのページおよびヘッダのデータ長が記録されている. またページやオプションページも含まれている. ページは標準ページ(すなわちデータ), そのファイル全体の情報をもつヘッダページ, そしてアクセスコントロールページのいずれかである. アクセスコントロールページはアクセスを制限する付加バイトが設けられている.

FTPを実装する際の問題は, FTP仮想ファイル形式をローカルファイルシステムの内部様式に対応づけるときに発生する. 例えばUnixシステムはデフォルトではファイル構造であり, 連続したバイト列である. Unixがレコード構造のファイルを受け取ると, ファイルのレコード構造を保ちながらその情報を内部にどのように格納するかを決めなくてはならない.

データ転送手続きでは2つのTCPコネクションを使用する. このとき普通はそれぞれの終端に2つのポート, そしてコネクションごとに1つのポートが必要になる. ユーザはそのポートを指定することもできるのだが, デフォルトもある. クライアント側では, デフォルトのデータポートはコネクションコントロールポートと同じである. サーバ側では, データポートはコントロールコネクションポートの隣(−1)である.

デフォルトポートの代わりに, ユーザは必要に応じて任意のアドレスを指定することができる. 例えば, 受取り側のデータポートは(普通はクライアント側にあるのだが)他のサーバにすることができる. コマンドによってファイルを最初のサーバから2番目のサーバに転送することになる.

データ転送は実際には3つのモードのいずれかになる. 普通はストリームモードが使われる. これはデータをほとんど, あるいはまったく処理せずに通過させる. このモードでは, 送出しているホスト側でデータストリームの最後にend of file(EOF)マークが付加され, データコネクションが閉じられて効率的である. ストリームモードでは, 他のファイル転送のためには再びコネクションを開く必要がある.

他の2つの転送モードはブロックモードと圧縮モードである.ブロックモードでは,データはヘッダが前についたデータブロック列として形成される．ヘッダはカウントフィールドとディスクリプタコードからなっている．ディスクリプタはファイルの最後のブロック，レコードの最後のブロック，リスタートマーカなどを識別する．

suspect データインジケータは本当の回復機構ではなく,磁気テープドライブからの読込み時にエラーを起こした場合などの，疑わしいデータをマークするのに使われる．リスタートマーカはエラー回復に用いられる．これはビット落ちやデータの混乱が問題になっているのではないことに注意してほしい.これらの問題は下層の TCP サービスによって扱われるものである.そうではなく，ここで問題となるのはホスト，FTP プロセス，あるいはその下のネットワーク(すなわちゲートウェイ)の誤りである．

ヘッダのリスタートマーカフラグは，そのブロックにはリスタートマーカがあることを示してくれる．そのマーカはデータの送り手にとって意味のあるものである．それはビットカウントやレコードカウント，あるいは磁気テープのラベルにもなり得るものである.受取り側の FTP プロセスはマーカの位置を見つけ，ユーザにその情報を教える．システム障害が起こった場合は，ユーザはその最後のマーカの位置を見て転送を再開させる．

圧縮モードはブロックモードに(妙な話だが)データ圧縮の付加的機能の付いたものと同じである.圧縮するときは繰り返されているある1文字(例えばスペース)を取り去り,バイト数とその文字で置き換える．繰返しがない場合は圧縮は行われない．データの繰返しには2種類考えられる．ホワイトスペースが最も一般的なタイプである．ホワイトスペースの繰返しは圧縮モードでは1バイトで表される.そのバイトの最初の2ビットは1で,6ビットのバイト数がそれに続く.64スペースまでなら，1バイトのデータで表せる．

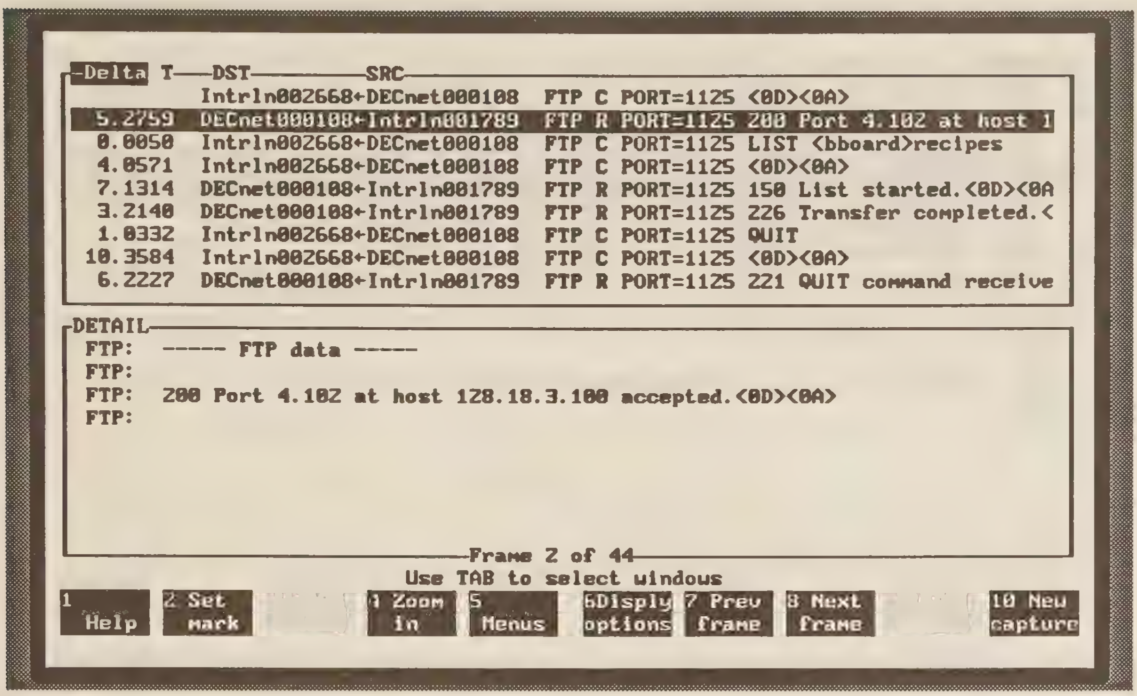
通常のバイトと圧縮したホワイトスペースをどのように区別したらよいのだろうか. netASCII のキャラクタセットでは,基本的キャラクタに限られるので,最初のビットは常に0である.そこで1ビット目が0ならば，これはキャラクタである.その2つのビットが1ならば，ホワイトスペース数である．

スペース以外のキャラクタの繰返しも圧縮することができる．最初の2つのビットがそれぞれ1と0ならば，これは他のキャラクタの繰返しであることを示している.これらの繰返しでは，全部で2バイトを必要とする．

その最初の2ビットの後の6ビットは，何文字(64まで)繰り返されているかが示されている．そのエスケープシーケンスの2バイト目は繰り返されるキャラクタである．

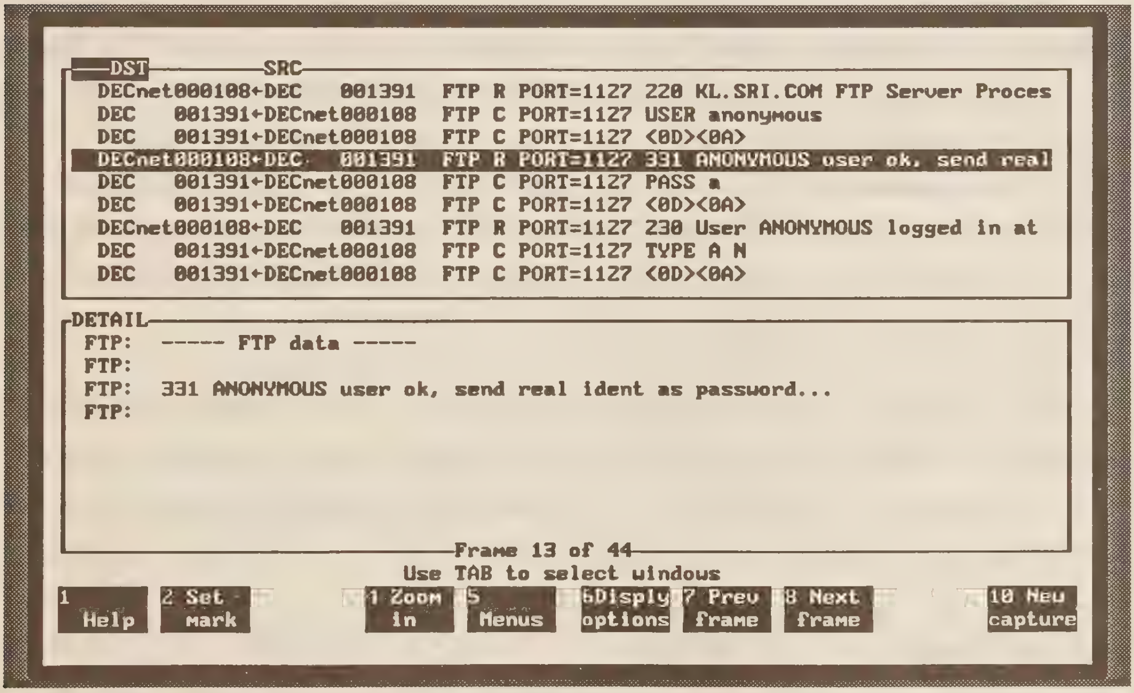
14.3.2 FTP データ

図 14-8 から 14-10 にネットワーク上の典型的な FTP 通信の様子が示されている．図 14-8 では，ホストはセッション要求をアクセプトしている．応答のテキストと同様に応答のコードにも



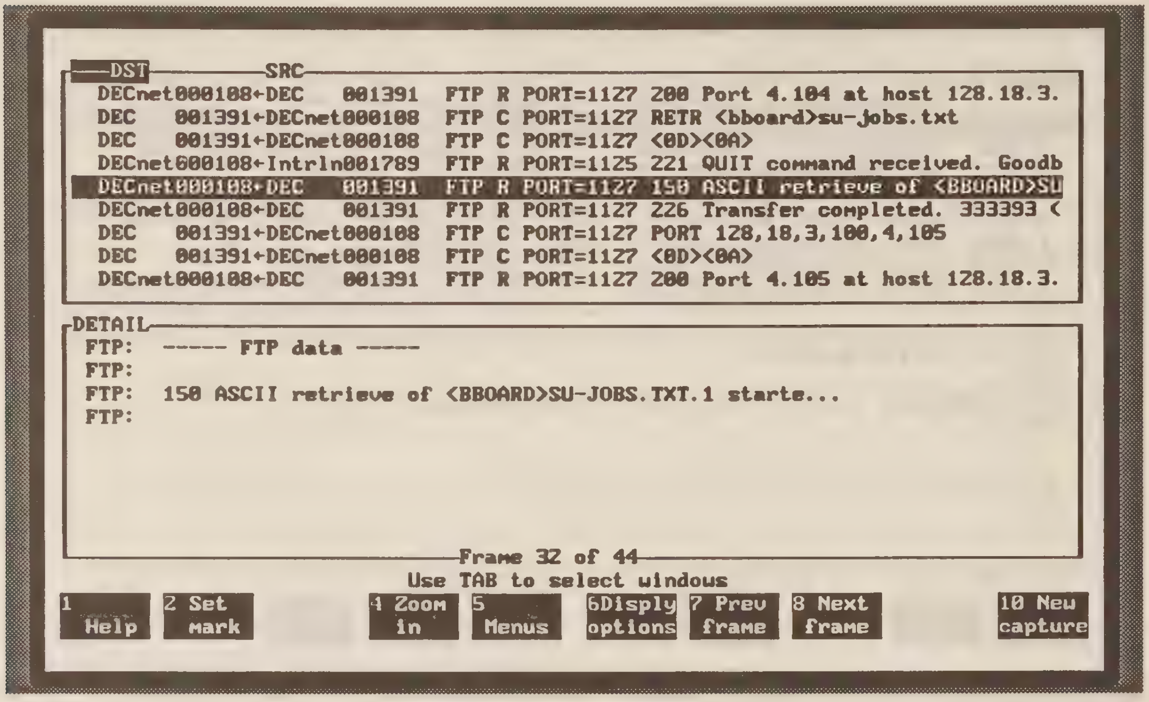
Courtesy of Network General

14-8 FTPセッションの開始



Courtesy of Network General

14-9 Anonymous FTP



Courtesy of Network General

14-10 ASCIIデータの転送

注意．応答コードも単純メール転送プロトコル(Simple Mail Transfer Protocol：SMTP)の議論で見たのと同じ方法に従っている．そのセッションはきわめて短く(<bboard>)，レシピのファイルリストからなっている．

図 14-9 には anonymous FTP として知られる慣習が示されている．その慣行では，公共のファイルにアクセスしたい場合には，その特定のホストにログインしなくてもよいことになっている．ログインする代わりに，ホストはユーザ名を Anonymous（ユーザ名は FTP とするときもある）とし，パスワードは何でも許されるが，そのお返しとしてパスワードフィールドにはその人の本当の名前を入力しておくことが望ましい．

最後に，図 14-10 はファイルの ASCII 読出しを示している．ユーザは RETR(<bboard>su-jobs.txt)コマンドを送った．ハイライトされたコマンドの直前に quit コマンドがあるのに注意してほしい．この quit コマンドは同じ DEC サーバに同時に発生する他の FTP セッションに対するものである．

14.4 FTAM

OSI ファイル転送，アクセスおよび管理(File Transfer, Access, and Management：FTAM)プロ

トコルは，目指すところは同じであるものの，様々に異なるファイル機構を念頭に設計されたものである．そのようなものとして，FTAM は一般的モデルを提供しているのだが，実装ではその中の特定のサブセットをとりあげている．このプロトコルのサブセットのとり方は非常に重要である．2 つの実装で同じサブセットやプロトコルをサポートしていなかったり，同じサブセットをネゴシエートできないなら，機能上完全に接続することは必ずしもできない．

FTAM の一般的ファイル格納モデルは 1 つのファイルで，データユニット列に分けられたものである．データユニットは木構造のノードで構成される．ノードには 0 個以上のデータユニットが付けられる．従来のファイルシステムでは，ノードがファイルで，データユニットはファイル内のデータであった．FTAM では，モデルは 1 つのファイルをベースにしている（ディレクトリ内には複数のファイルの概念はない）．FTAM モデルでは，データユニットはレコード，ページ，ブロックあるいはファイル内の他のすべての構造のユニットである．

この一般的モデルに従って，ほとんどのファイル構造を作ることができる．例えば，順テキストファイルはルートノードの下の 1 つのノードからなり，レコードごとに一連のデータユニットがつけられる．バイナリファイルでは 1 つのノードと 1 つのデータユニットから構成される．

Btree や索引順アクセス法(Indexed Sequential Access Method：ISAM)などの階層的ファイル構造は，複数レベルの木構造として作られる．この潜在的には無限の木構造に順序づけをするのに，FTAM モデルではデータユニットをとり，ファイルアクセスデータ単位(file access data units：FADU)を構成させていく．FADU は葉のノードに至るまでの木構造の一部分である．このため，全ファイルは 1 つの FADU とも考えられ，より低いレベルの部分も同様に FADU になれる．

オペレーションの FADU モデルにより，仮想ファイルに対する操作を構成することが概念的にやさしくなる．ファイルに対する実際の操作は，ローカルファイルに対する操作によって実行されることを思い出していただきたい．FTAM プロトコルで必要なことは，どの操作を実行する必要があるかを通信する能力である．

FTAM セッションは一連のネストされたレジュメによって構成される．これらのレジュメには実行すべきオペレーションが書かれている．例えば，association レジュメはファイルを選択するのに必要である．選択した後で，ファイルはオープンされ，すべての転送の操作はその後で行われる．アクセスは順次的であることに注意してほしい．FTAM は DAP と同様に，ファイル処理に対しては非同期モデルは提供しない．

一般的 FTAM モデルはサービスクラスと呼ばれる機能モジュールに分解される．サービスクラスはプロトコルの能力によってサブセットに構造化していく．サービスクラスはプロトコル内の能力のレベルで，接続しているときのコネクションの両端によってネゴシエートされる．FTAM には 5 つのサービスクラスが明示されている．

- “ファイルアクセス” はレコードレベルのアクセスと操作を可能にする．

・ファイルの全体や一部分の“ファイル転送”.
・“ファイル管理”：ファイルの属性の読出しと変更.
・“ファイル転送と管理”はファイルの作成および消去も含む.
・“無制約”の場合はサービスクラスを使わずに，利用可能な機能ユニットをベースにしてネゴシエーションが行える.

機能ユニットはプロトコルの能力を分割するより細粒度の方法である．サービスの特定のクラスでは，機能ユニットには必須のものと省略可能なものがある．サービスのクラスではオプションの機能ユニットに対するネゴシエートをいつも行うことができる．無制約のサービスクラスは必須の機能モジュールなしのものである．以下のものはFTAMで定義されている10の基本的な機能ユニットである.

・カーネル：コネクションを張るための基本的ファイルサービス
・読出し
・書込み
・ファイルアクセス：ファイル内のFADUの位置指定と操作
・制限つき管理：ファイルの作成と削除，属性の読出し
・拡張ファイル管理：ファイルの属性の変更
・レジュメが効率的に作成されるような操作の“グループ分け”
・回復：選択した，あるいはオープンしたレジュメの再現
・データ転送の再開
・FADUのロック

機能ユニットとサービスクラスはプロトコルで利用可能なサービスを扱っているのだが，FTAMのレジュメで取り交わされる仮想ファイルの中にどのようなデータタイプが含まれるかが文書種別によって制限されてしまう．各ユーザはそれぞれのファイルのタイプを，データユニットの構造とFADUを仮想ファイル格納モデルの中から特定して，適切な権限を使って登録しなければならないこともある.

ユーザ定義の文書種別に加えて，ISOは以下の5つの基本的な文書種別を定義している.

・FTAM-1：構造なしのテキストファイル
・FTAM-2：順次テキストファイル
・FTAM-3：構造なしのバイナリファイル
・FTAM-4：順次バイナリファイル
・FTAM-5：単純な階層的ファイル

FTAMモデルはシングルファイルを基にしている．ほとんどのオペレーティングシステムは複数のファイルをもち，ディレクトリとして構造化されている．国家標準技術局（NIST，以前はNational Bureau of Standardsとして知られていた）で開発された標準は，補足的な文書種別を用

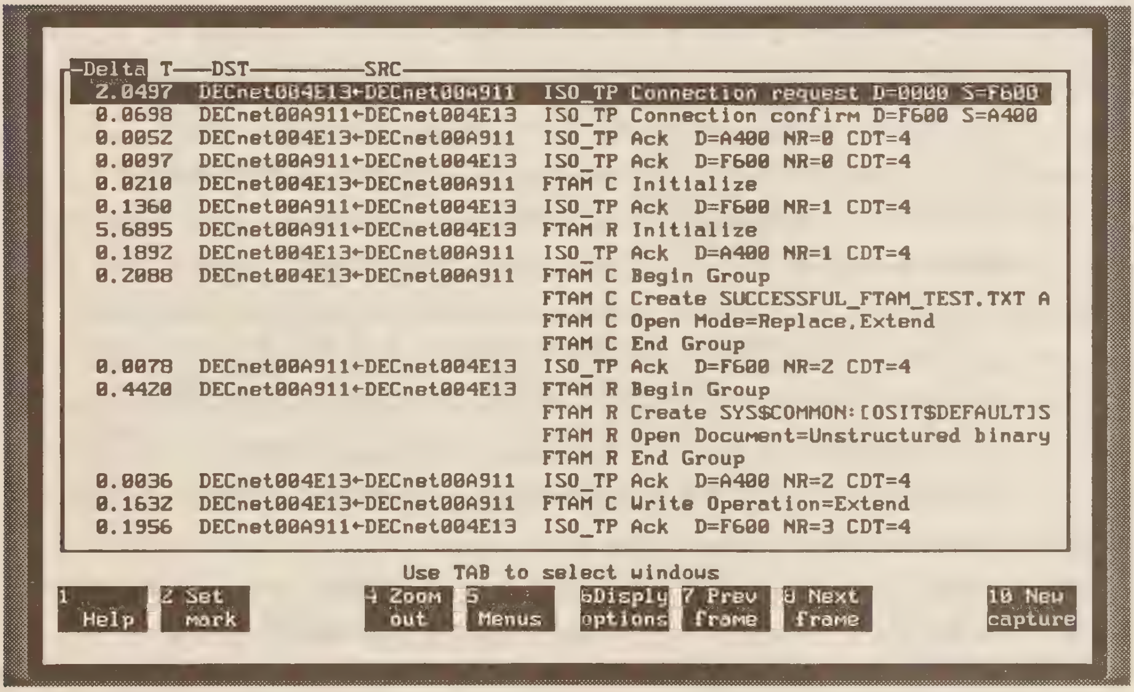
意しており，それはディレクトリ内のファイルを特別な種類のファイルとして扱うものである．ユーザはこの特別ファイルを取り出し，それを個々のファイルに移動することができる．

図 14-11 は典型的 FTAM セッションの例を表している．最初に起こっていることを見ると，OSI トランスポートプロトコルの仮想回線がつながれ，コネクション要求と確認のパケットが受理されている．

その回線がいったん初期化されると，FTAM 初期化メッセージが送られるが，それはおそらく OSI ACSE，プレゼンテーション，そしてセション層の初期化メッセージの中に埋め込まれて受け取られることになるだろう．もちろん，すでにあるトランスポート層のコネクションを再使用するようにセション層にさせることも可能である．その場合には初期化は不要である．この初期化メッセージは下層のトランスポート層でもまた，初期化応答として受け取られることになる．

FTAM association がいったんうまくいけば，要求しているノードは一連のオペレーションを構造化してまとめて送出する．まずファイルが作成され，そのファイルは“replace,extend”モードでオープンされる．replace,extend モードは FTAM 文書種別の 1 と 3 で使用される．文書種別 2 では，代わりに挿入モードが使用される．

次に，相手のノードは各オペレーションに対する確認メッセージを送り返す．返ってくるファイル名は，構造なしのバイナリドキュメントタイプをオープンするオペレーションに変えられたフルネームであることに注意すること．そして最初のノードは戻ってファイルを拡張する．



Courtesy of Network General

14-11 FTAM トラフィック

図 14-12 は FTAM の初期化要求をより詳細に表している．そのサービスは機能ユニットを基に作られてきた．そのノードは write を要求し，ファイル管理を制限し，グループ化をしてきた．属性グループはどのファイル属性がセッションに対してネゴシエートされるかを示す．

FTAM サービス品質(quality of service)はこの操作に対してリカバリが必要かどうかを示すのに使われる．リカバリや下位のチェックポイント機能を使うとセッションが大変に遅くなってしまうので，アプリケーションが必要としないときは，しばしば無効にされる．

この初期化要求は構造なしと順次テキストとの 2 種類の文書種別が使用できることを示している．実際のファイルシステムの中でそれらの文書種別が何を意味するか，それらがいかにしてアプリケーションによって解釈されるかは，通信している両側のノードに依存する．

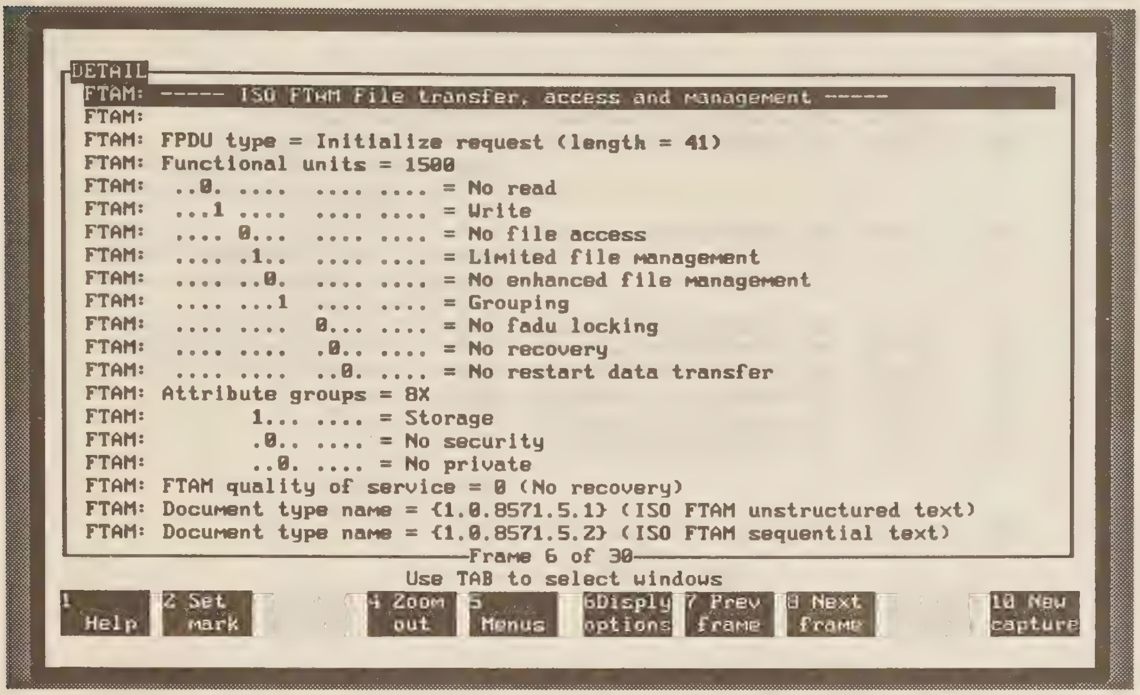
図 14-13 はファイルをクローズして選択からはずす操作について，もう少し詳しく示している．それぞれの FTAM コマンドの間に，可能なプレゼンテーションコンテクスト識別子の中で，今度のコマンドに適用するのはどれかを示すプレゼンテーション層ヘッダが入っている．

図 14-14 はデータ転送オペレーションを扱ったセッションを表している．まずファイルが作られ，そしてそのファイルは拡張される．次に 160 バイトのデータが転送され，data end(データレジュメの終わり) が送られ，そして転送が終了する (転送レジュメの終わり)．最後に，そのファイルはクローズされ，選択からはずされて，他のユーザのために解放される．

図 14-15 はファイルを構造なしテキスト文書として作り，オープンするもう 1 つの簡単なセッションを示している．送られたテキストデータは 10 個あり，それらは下のイーサネットではすべて同じパケットに含まれている．テキストデータは data end メッセージと転送終了レジュメが後に続く．FTAM の興味深い一面は，定義されてきたファイル属性の多さである (図 14-16 参照)．属性は 4 つのグループに分けられ，すべてのグループがサポートされる必要はない．

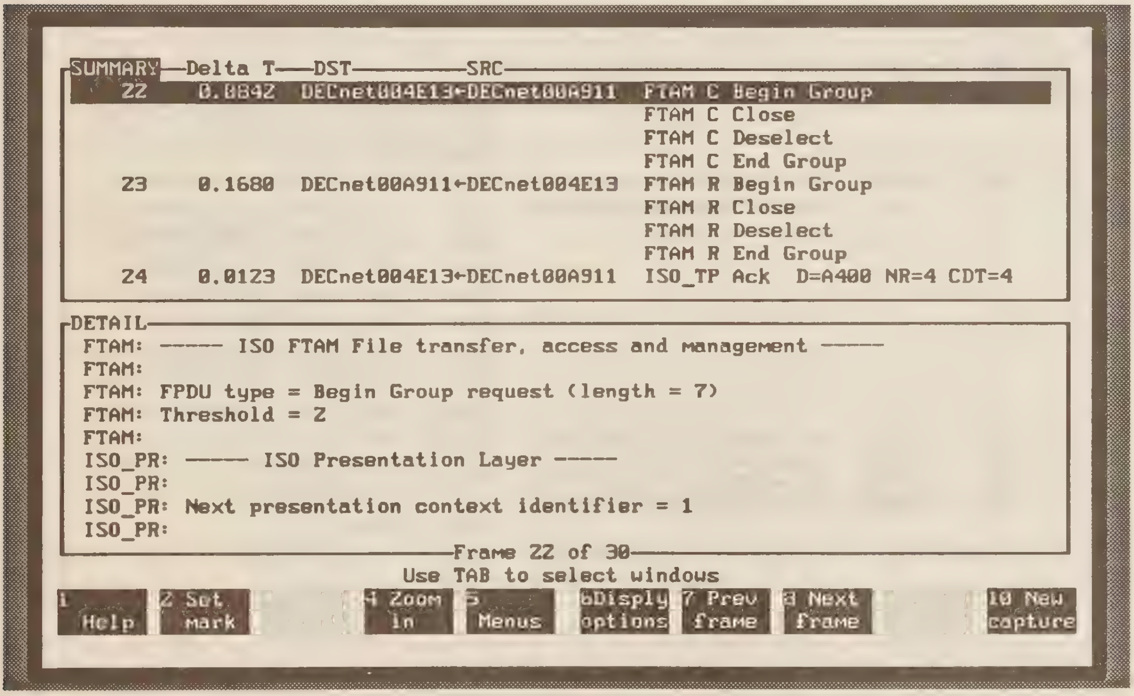
Kernel にはファイルの名前やアクションのパーミッションを含む，基本的な情報が含まれている．その内容の型はファイルの文書種別(あるいは制約セットと呼ばれる，より一般的な分類)を示す．この情報があるからといって，オペレーティングシステムが必ずしも他のユーザにそれを公開するわけではないことに注意されたい．Storage グループにはアカウントの情報が含まれており，ファイルアクセスに対してアカウントに課金されるかもしれない．Storage グループはデータと時間の情報と同様に，サイズや利用情報もまた示している．Security グループは様々な目的に使われる．そのアクセス制御はファイルのデータを見ることができる(または書込みのような他のオペレーションを実行できる)のは誰かを表す．法的制限はデータがどのように使用されるかを表す著作権のような情報を示している．

最後に，Private グループというものがあり，それは 1 つの固有属性しかない．これは実装やコンピュータ環境によって，OS 固有のファイル属性や情報を書き込む場所で，FTAM ではサポートされていない．



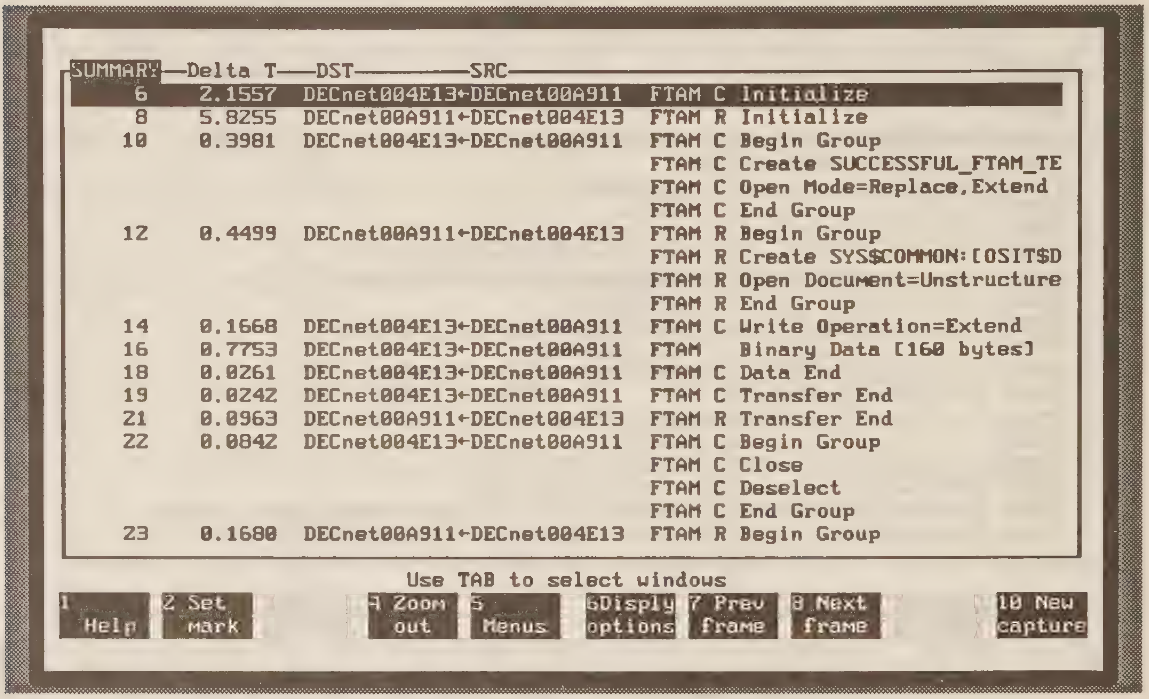
Courtesy of Network General

14-12 初期化要求



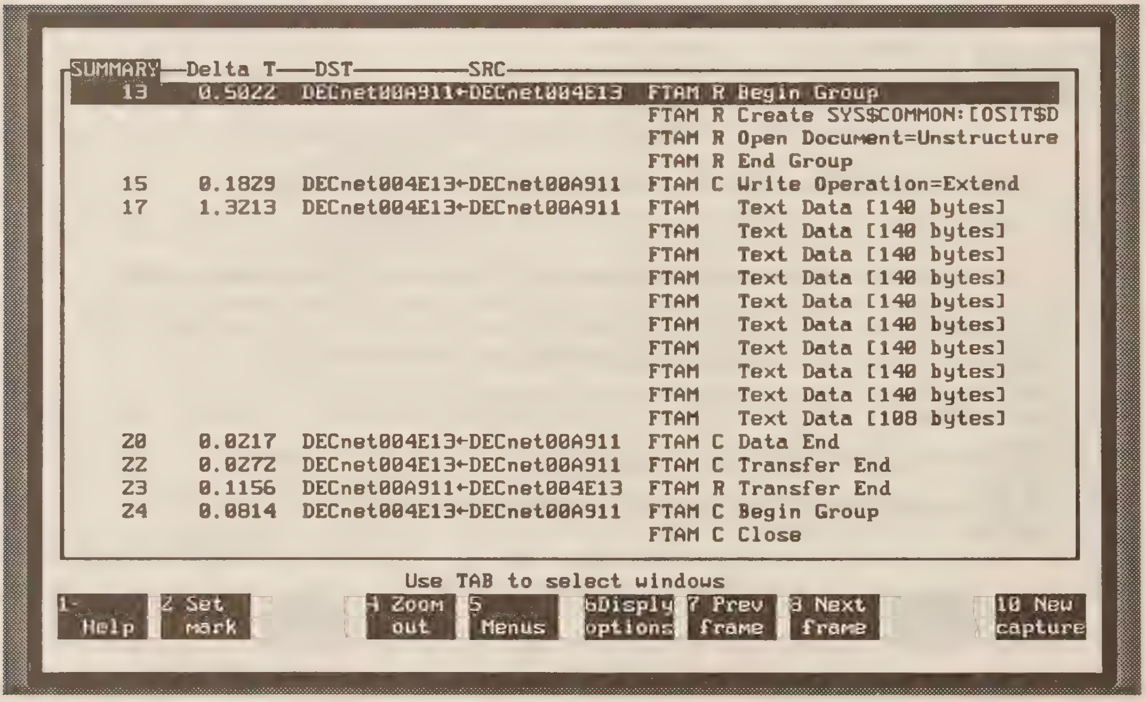
Courtesy of Network General

14-13 CloseおよびDeselect操作



Courtesy of Network General

14-14 データ転送要求



Courtesy of Network General

14-15 ファイル属性

Kernel グループ	ファイル名
	許可アクション
	内容の型
Storage グループ	ストレージのアカウント
	日時属性
	ID
	ファイル有効性
	ファイルサイズ
	予定ファイルサイズ
Security グループ	アクセス制御
	法的制限
	カレントアクセスパスワード
Private グループ	固有属性

14-16 FTAM 属性

14.4.1 Sun の FTAM 実装

Sun も他のほとんどのベンダと同様，FTAM をゲートウェイとして，その環境の中にすでに採り入れている．SunLink OSI は基本 FTAM を実装している．ゲートウェイはネットワークの端に位置し，OSI ネットワークへの FTAM セッションを行うことができる．ユーザはゲートウェイシステムを通じて OSI ベースのサービスにアクセスするだけである．16 章の「相互運用性」では，このことについてもう少し詳しく議論することになる．

14.5 資源の位置

ネットワーク上で資源を発見することは，徐々に難しいことになりつつある．この問題はこの本の中のいくつかの箇所で議論されている．15 章の「ネットワーク管理」では，SunNet マネージャの一部として，発見ツールについて説明する．9 章の「名前」では 3 つのディレクトリサービス (NIS+，NIS，そして DNS) について見て，さらに 2 つ (X.500 と DEC DNS) について触れた．ここで，さらに 2 つのサービスを見てみることにする．これらは人を探す問題に特に適している．電話番号，電子メールアドレスあるいはファックス番号のようなアドレスが見つからなくて人と連絡できないときほどいらいらするときはない．

ついに，X.500 のようなサービスのおかげで人を見つける問題の解決策が提供されるようになった．しかしながら，何年もの間，whois や finger サービスは基本的な知識の源泉となってきた．

14.5.1 whois プロトコル

ドメイン名前システムはサーバIPアドレスに対するドメイン名のように，マシンと相性のよいものを解決するのには優れているが，多くの人々にとってはやや低レベルである．whois サービスはネットワーク情報センター（ホストの SRI-NIC はこのサービスを提供していて，IP アドレス 26.0.0.73 または 10.0.0.51 で稼働中）から TCP サービスとして提供されているものである．単純な要求/応答プロトコルにより，このサービスは検索パラメータを受け，次の4つの情報を返す．

- 氏名
- 合衆国メーリングアドレス
- 電話番号
- 電子メールアドレス

データベースに登録するのは，各個人までできる．国防省通信局は Internet の DoD の部分を稼働させているが，ネットワークを通じてトラフィックを流すすべての各個人に登録を要請している．ユーザは次のところに電子メールを送る．

registrar@nic.ddn.mil

メールは氏名，2番目のイニシャル，合衆国メーリングアドレス（住所：メールストップ，略語，頭辞語の説明を含む），ZIP コード，電話，ネットワークのメールボックス1つを付記する必要がある．

サービスにアクセスするときには，ユーザは1つの名前を指定する．名前の指定が疑問符の場合は，サービスは可能なすべてのフォーマットを返してくる．

14.5.2 finger プロトコル

多くの Unix システムには，ネットワーク全体を通してユーザやホストの情報を見つけ出すのに使われる finger コマンドがある．尋ねるプログラムはリモートユーザ情報プログラム（remote

SMITH	Smith という名前や肩書を探す．
!SRI-NIC	SRI-NIC という肩書のみ探す．
.Smith	ラストネームが Smith の人を探す．
Mal...	Mal で始まるものをすべて探す．
smith@	知らないホスト上の Smith を探す．
@host	ホスト上のすべてのユーザを調べる．
smith@host	ホスト上のユーザ名 Smith を探す．
*SRI-NIC	SRI-NIC で終わるすべての名前を見つける．

14-17 whois サービス

user information program : RUIP)として知られている．RFC 1196 はユーザ（すなわち finger コマンド）と RUIP 間で使われるプロトコルを定義している．

このプロトコルは TCP のポート 79 を使う．ユーザはこのポートにコネクションを設定し，それを設定すると，信号が RUIP に送られて要求を処理する準備が整う．そのプロトコルは 1 行の質問からなる．RUIP は情報を回答し，コネクションをクローズする．この質問で転送されるデータはすべて単純な ASCII データである．

finger は 2 つの種類の質問が出せる．最初の種類では，リモートユーザの名前で質問する．それに対しては，ユーザに関する情報が返ってくる．2 番目の形は少し変わっていて，質問を回したいホストのリストからなり，それは発信元経路指定の形になる．普通は，質問は次のようになる．

```
myhost%   finger user2@host2
```

この結果，host2 とのコネクションが張られ，finger コマンド "user2" が送られる．その返答はユーザに関するホスト依存の情報になる．ワード(/w)フラグを付けると，もっと詳しい情報が送られる．

この質問の特別な形はユーザ名をヌルにすることである．これはオンラインのすべてのユーザについての情報を問うていることになる．リモートの RUIP は返答するか否かを選択できる．

ホストの探索は変わった形をしている．コマンドはリストの最初のホストに送られ，そのホストはその質問を他の RUIP に転送するように求められる．最初の RUIP はそれを受け付けるか，拒否するかのどちらかをしなくてはならない．単に無視することは許されない．

受け付けられると，その質問は最後のホストがそれを受けるまで転送されていく．そして，その結果は送り返されてきて返答がユーザに示される．

質問が受け付けられたとき，その返答は少なくともユーザの人名は含んでいる．端末の場所，オフィスの場所，オフィスの電話番号，仕事のテーマ，ユーザアカウントのアイドル時間の量(そのユーザが端末で実際に仕事をしているのか，そのスロットを占めているだけなのかが示される)などの他の情報が含まれることもある．

ほとんどのシステムではさらに各ユーザが管理できる.plan ファイルを含める．そのファイルの中にはユーザが外の世界に示したい情報が書かれている．それにはユーザが仕事をしている現行プロジェクトも書けるが，特別な情報も同様に書くことができる(すなわち，"オフィスにいなければ家に電話しないで下さい．" など)．

システムに WilliamA と WilliamB がいたときに，ユーザ名 William を送る場合のように，受け取るユーザ名は不完全でもよい．それは RUIP に任されている．その要求は不完全であるといって拒否することもできれば，それより長い名前をすべて探したり，入力に soundex を走らせて，ユーザ名と与えられた文字列が「似て聞こえる」かを調べることも可能である(入力した名前のスペルが間違っている場合など，有用な特長である)．

finger プロトコルは，その定義づけから，セキュリティ上危険を伴う．外の世界に自分の環境の

ことを伝えているのだから，fingerが少しでも動くべきか，そして動くのだったらどんな情報を返答に入れるのかを決めるのは，ローカルネットワーク管理者あるいはそのホストの管理者である．

しばしば表示される情報は，最後のメールメッセージがそのユーザに読まれた時間と，メール

```
rutgers$ finger weather@madlab.sprl.umich.edu
(madlab.sprl.umich.edu)
Login name: weather                    In real life: Home of the Weather Re-
port
Directory: /sdm/weather                Shell: /bin/false
Never logged in.
New mail received Thu Nov 15 15:53:05 1990;
  unread since Fri Sep 14 13:39:56 1990
Plan:

614
FQUS1 KDTW 130325
MIZ028030-131010-

DETROIT METROPOLITAN AREA FORECAST
NATIONAL WEATHER SERVICE DETROIT MI
1030 PM EST SAT JAN 12 1991

.TONIGHT...VARIABLE CLOUDINESS. COLDER...LOW AROUND 20. LIGHT
AND VARIABLE WIND.
.SUNDAY...MOSTLY SUNNY. HIGH IN THE MIDDLE 30S. WIND BECOMING
SOUTHWEST 5 TO 15 MPH.
.SUNDAY NIGHT...INCREASING CLOUDINESS...LOW IN THE MIDDLE 20S.
.MONDAY...CLOUDY WITH A 30 PERCENT CHANCE OF SNOW. HIGH IN
THE MIDDLE 30S.

GHW

rutgers$ finger weather@groucho.ucar.edu
(unidata.ucar.edu)
Login name: weather                    In real life: Denver Metro Forecast
Directory: /data/weather               Shell: /data/weather/dbq
Last login Fri Jan 11 13:28 on ttypb from CSLI.Stanford.ED
No unread mail
Plan:

914
FQUS1 KDEN 130435
COZ011-131100-

DENVER METROPOLITAN FORECAST
```

14-18　Fingerサービス

が読まれずに待っているかどうかである．システムによっては，最後に受け取ったメールが誰からかを示してくれさえするものもある．この情報は会話の流れ(メールが読まれているか)を追うのに使われる．

クライアント側にもセキュリテイ問題が起こり得る．意地の悪い finger プログラム(あるいは .plan ファイルで意地の悪いことをするユーザ)がいて，エスケープシーケンスのようなものを入れておいてユーザの画面を混乱させることもできてしまう．よくできた finger クライアントだと，表示できないキャラクタにユーザがフィルタをかけて取り除いてしまうこともできる．

finger プロトコルの最後の注意は重要な点を示唆する．それはユーザは必ずしも人間でなくともよいということである．finger プロトコルには人間でない実体の情報をもたらす実装もある．例えば，finger プロトコル上の天気用に作られたサーバは，weather@madlab.sprl.umich.edu や weather@groucho.ucar.edu など，少なくとも 2 つのコンピュータサイトにある(図 14-18 参照)．

finger プロトコルを使って，現在の内容物をネットワークを通じて知らせてくれるコークマシンもあるといううわさもある．finger RFC の作者である Rutger の David Zimmerman によれば，「CMU のコークマシンには伝説があって——coke@???.cmu.edu (???の名前は忘れちまった)に finger をかければ，その内容物の情報が得られるぞ．」

14.6　自明 FTP

自明(Trivial) FTP (TFTP)は FTP の簡易版で，リモートサーバとファイルの読み書きをやりとりするのに使われる．TFTP はメールメッセージをあちこちに移動させるのに使われることもあり，そしてディスクレスノードがネットワークを通じてオペレーティングシステムを得る方法にもなっている．

TFTP のデータ転送には 3 つのモードがある．

・netASCII

・Octet

・ファイルではなくユーザへの netASCII (例えばメール)

すべての転送はファイルへの読み書き要求による．この要求により，コネクションも張られる．サーバがその要求に答えると，データ転送は 512 バイトのブロックで行われる．これよりも少ない(0 から 511 バイト)ブロックは転送終了を表す．

コネクションの要求者は読出し要求(read request : RRQ)や書込み要求(write request : WRQ)を送出し，受理を受け取る．それに従って，各データパケットはブロック番号を得る．ブロック番号は 1 で始まる連続数である．ブロック番号 0 は最初の書込み要求の承認パケットである．

すべてのデータブロックは明示的に ACK される．ステップ的に承認をとることで，簡単なプロトコルを実現している．エラーのほとんどはエラーパケットを通じて中断を引き起こす．エラー

パケットはACKされない.

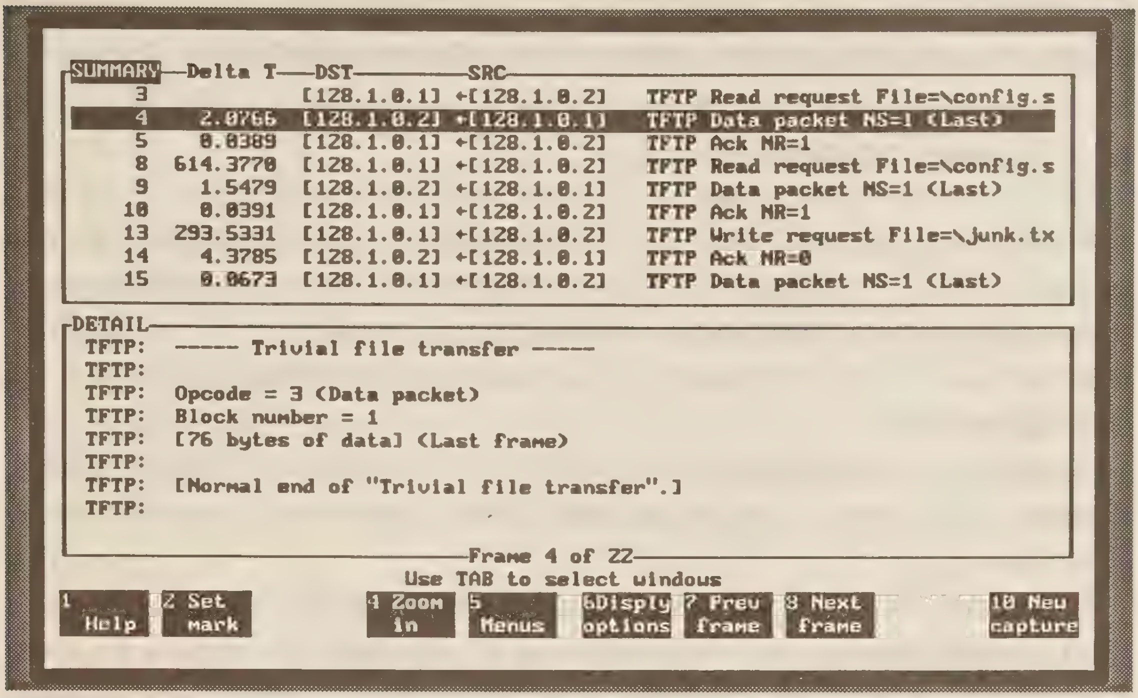
TFTPはUDPを基にしており,有名なポート番号69を使う.そのコネクションはUDPレベルのトランザクションID(TID)によって識別される.初期要求は発信元のTIDをもっていて,10進数で69(8進数で105)の周知のポートに送られる.この初期要求に対する応答はサーバプロセスによって選ばれた,違うTIDをもつことになる.結果としての発信元/宛先TIDは次のやりとりで使われる.TIDはランダムに選ばれるので,近接したものに同じ番号が振られることはめったにない.

簡潔さ？　それがこのアイデアのすべてである.そこには4つのパケット種別しかない.読出し,書込み要求で流れをつくる.データパケットと確認応答でデータを転送する.フルブロック未満のブロックでセッションを終了する.この簡単さのためにTFTPがリモートブートに使用されるのである.それは大変簡潔なので,ROMチップにプロトコルを入れてしまうことができる.

図14-19はTFTPプロトコル全体を示しており,4つの異なるパケットから構成されている.図14-20と図14-21はネットワーク上のあるTFTP通信を表している.図14-20では,やりとりはconfig.sysファイルに対する簡単なファイル読出し要求によって始まっている.明らかに,これは

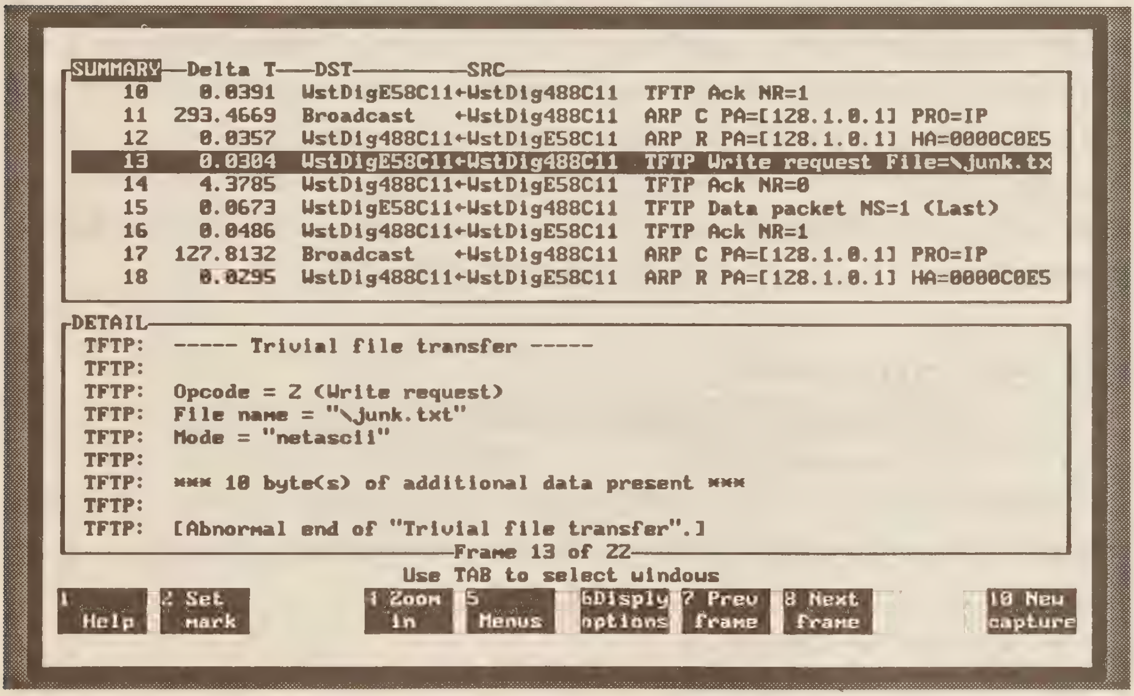
Read Request (RRQ) と Write Request (WRQ) パケット	
命令コード	rrq = 1 (2bytes)
	wrq = 2
ファイル名	
終端	1 バイト = 0
モード	モードはnetascii, octet, mailのいずれかである．mailの場合はファイル名はユーザ名となる．mailモードは常にWRQである．
終端	1 バイト = 0
データパケット	
命令コード	data=3
ブロック番号	2 バイト
データ	512 バイト：通常パケット
	EOT: 0-511 バイト
ACK パケット	
命令コード	ACK = 4
ブロック番号	(2バイト)
エラーパケット	
命令コード	命令コード = 5
エラーコード	2 バイト
エラー文字列	
終端	1 バイト = 0

14-19 自明FTPパケット



Courtesy of Network General

14-20　TFTP転送の最後のパケット



Courtesy of Network General

14-21　TFTP書込み要求

PCが自分をブートしているところである．図14-21は簡単な書込み要求を示している．TFTP書込み要求が行き渡る前に，TFTPサーバのIPアドレスを探し出すARP要求が入っている．

14.7　その他のTCP/IPサービス

1983年のRFCでJon Postelによっていくつかの有用なユーティリティが定義された．echo serverはTCPのポート7を見続けるTCPベースのサーバである．コネクションが設定されると，サーバに送られるデータは呼出し側のユーザがコネクションを終わらせるまで，すべてのデータが送り返される．そのサービスのUDPバージョン(これもポート7を見続ける)はデータグラムを送り返すだけである．

discardプロトコルは受け取ったデータをすべて捨ててしまう．TCP(ポート9)をベースにした場合はコネクションがアクティブである限り続き，UDPポート9の場合には1回のデータグラムのみに行われる．

ポート19のキャラクタジェネレータは，入力にかかわらず何らかのデータを送る．データは何でもよいのだが，認識できるパターンであることが望ましい．一般的には，そのプロトコルは1行に95のプリント用ASCIIキャラクタを受け，1行72キャラクタにして繰り返しパターンを作ってプリントする．

daytimeプロトコル(ポート13)は日付と時間をASCII形式で表示する．daytimeはユーザにわかりやすい時間を示すのに対し，timeプロトコル(ポート37)はマシンの解釈しやすいバージョンを送るものとしての意味をもっている．timeは1990年の1月1日の午前0時からの秒数を32bitで表したものである．

daytimeとtimeプロトコルはNetwork Time Protocolの基本的な前身である．古いプロトコルではそのマシンでの時刻が提供されるが，NTPでは多くのコンピュータの時計を上位の参照先と同期させている．

14.8　ラインプリンタデーモン

ラインプリンタ(line printer：lpr)はバークレーで開発されたコマンドセットで，遠隔プリンタにジョブをサブミットしたり，プリントキューを管理したりするためのものである．そのプロトコルはクライアントマシン上の4つのユーザコマンドと，サーバ上のラインプリンタデーモンからなっている．4つのユーザコマンドは次の通りである．

・lpr：ジョブをキューに割り当てる
・lpq：キュー中のジョブを表示する
・lprm：ジョブをキューから取り除く

・lpc：キューをコントロールする

このプロトコルは，ジョブをどのようにプリントするかまではコントロールしないことに注意されたい．lpr コマンドのすることはローカルプログラムが見つけやすいところにジョブを置くことである．

lpr デーモンは TCP に基づくサービスで，ポート 515 を見続けている．そのプロトコルは大変シンプルである．それぞれのコマンドは 1 バイトのコマンドコードで始まり，プリンタキューの ASCII 名が続き，何らかのオペランドが後にくる．

ほとんどのコマンドは大変簡単な要求/応答プロトコルで，実行されればコネクションの切断が行われる．図 14-22 は LPR のために定義されたコマンドである．

その手続きはまずプリントオプションとコマンドを含むコントロールファイルを転送することからなる．次に，データファイル(プリントすべきデータ)が転送される．コントロールファイルはすべて DFA という文字，3 桁の数字，そしてサブミットするホストの名前で始まる名前でなければならない．

コントロールファイルは一連のコマンドで成り立っている．各行は 1 キャラクタのコマンドとオペランドからなる．図 14-23 は使用可能なコマンドを示す．それらは状況に依存することに注意されたい．このファイルは少なくとも 3 行以上でなければならない．H コマンドは依頼したホストを示し，P コマンドは依頼したユーザである．出力するには少なくとも 1 つ以上の小文字のコマンドが入っていなくてはならない．

これ以上は簡単にならない．

コード	説明
1	待ち状態のジョブをすべてプリントする．
2	ジョブを受け取る．
3	キューの状態を送る(短縮リストで)．
4	キューの状態を送る(省略なし)．
5	ジョブを取り除く．
6	ジョブのサブステートを取り除く．
ジョブサブコマンドの受領	
1	中断する．
2	コントロールファイルを受け取る．
3	データファイルを受け取る．

14-22 lprコマンド

コード	名前	説明
C	表題ページの Class名	表題ページにプリントするクラス名を設定する．名前は 31 文字以下のバイト列．名前は省略可能．その場合は，ファイルをプリントするホストの名前が使用される．このクラスは，そのプリントジョブを発行するホストを表示するのが慣習である．表題プリントコマンド(“L”)とともに使われないと，無視されてしまう．
H	ホスト名	プリントジョブのソースとしてのホスト名を指定する．このコマンドは コントロールファイル内に含まれていなければならない．ホスト名は 31 以下のバイト列．
I	インデントプリント	“f” コマンドでプリントするファイルに対して，与えられたカラム数だけインデントする(他の出力生成コマンドでは無視される)．インデントのカウント数はすべて10進数．
J	表題ページの ジョブ名	表題ページにプリントされるジョブ名を設定する．ジョブ名は 99 文字以下のバイト列．省略可能．ジョブ名は"プリントされた"ファイルの名前が使われるのが慣習である．表題プリントコマンド(“L”)とともに使われないと，無視されてしまう．
L	表題ページの プリント	表題ページをプリントする．ユーザ名は省略可能．「表題ページの Class名」と「表題ページのジョブ名」コマンドを有効にするにはコントロールファイル内に前もって書かれていなければならない．
M	プリント後に メール	プリントのオペレーションが終了後，“H” エントリで指定されたホストの，オペランドとして与えられたユーザにメールを送る(成功，不成功)．
N	ソースファイル名	データファイルの生成元のファイル名を指定する．ファイル名はキューにたまり，タイトルがなければ “p” コマンドでプリントされる．131 以下のバイト列．
P	ユーザ ID	プリントジョブを発行するエントリのユーザ IDを指定する．このコマンドはコントロールファイルに書かれていなければならない．ユーザ ID は 31 文字以下のバイト列．
S	シンボリック リンクデータ	Unix システムのシンボリックリンクデータを記録するのに使われ，ファイルのディレクトリエントリをプリント後に変更しても，その新しいファイルをプリントしてしまわないようにする．データファイルがシンボリックリンクで張られていない場合は無視される．
T	pr 用のタイトル	“p” コマンドでプリントするファイルのタイトルをつける(他のプリントコマンドの場合は無視する)．タイトルは 79 文字以下のバイト列．
U	データファイルの Unlink	指定したファイルがもはや不必要であることを示す．データファイルのみに適用．
W	出力幅	“f”，“L”，“p” コマンドに対して指定されたカラム数で出力を制限する(他の出力生成コマンドの場合は無視される)．幅数のオペランドはすべて 10進数．小さな値に置き換えられることもある．幅のデフォルト値は 132.
1	Troff R フォント	troff R のフォントのファイル名を指定する．デフォルトでは Times Roman でプリントされる．
2	Troff I フォント	troff I のフォントのファイル名を指定する． デフォルトでは Times Italic でプリントされる．
3	Troff B フォント	troff B のフォントのファイル名を指定する．デフォルトでは Times Bold でプリントされる．

14-23 LPRコントロールファイル

コード	名前	説明
4	Troff S フォント	troff S のフォントのファイル名を指定する．デフォルトではSpecial Mathematical Fontでプリントされる．
c	CIF ファイルのプロット	データを CIF (Caltech Intermediate Form)グラフィックス言語として扱い，データファイルをプロットする．
d	DVI ファイルのプリント	データを DVI (TeX 出力)として扱い，データファイルをプリントする．
f	フォーマットファイルのプリント	データファイルを単なるテキストファイルとしてプリントする．必要に応じて改ページを入れる．HT，CR，FF，LF，BS 以外のコントロールキャラクタは無視される．
g	ファイルのプロット	データを Berkeley Unix plot ライブラリからの出力として扱い，データファイルをプロットする．
k		Kerberized LPRクライアントとサーバ用に予約．
l	コントロール文字を変換せずにプリント	指定したデータファイルを，コントロールキャラクタにフィルタ(“f” コマンドのように) をかけずにプリントする．
n	ditroff 出力ファイルをプリント	データを ditroff 出力として扱い，データファイルをプリントする．
o	PostScript 出力をプリント	データを標準 PostScript として扱い，データファイルをプリントする．
P	“pr” フォーマットでファイルをプリント	データファイルをヘッダ，ページ数，ページ付けを付けてプリントする．ヘッダにはプリント開始時の日時，タイトル，ページ番号識別子とページ数が含まれる．タイトルは “T” (Title) コマンドが与えられなければ，“N” コマンドで指定されたファイル名になる．テキストのページがプリントされると，新しいページが新しいページ数で開始される(ページの長さを指定する方法はない)．
r	FORTRAN キャリッジコントロールでプリント	各行の最初の列を FORTRAN キャリッジコントロールとして解釈してデータファイルをプリントする．FORTRAN の標準ではここは空白，“1”，“0”，“+” のキャリッジコントロールに制限されている．FORTRAN プログラマの多くは “-” (トリプルスペース) もうまくいくように期待する．
t	troff 出力ファイルのプリント	データファイルを Graphic Systems C/A/T フォトタイプセッタ入力としてプリントする．Unix troff コマンドの標準出力．
v	ラスタファイルのプリント	Sun ラスタフォーマットファイルをプリントする．
z	Palladium	プリントシステムのための予約．

出典 : RFC 1179

14-23（続き） LPRコントロールファイル

より進んだ勉強のために

K. Harrenstein, M. Stahl, E. Feinler, *NICNAME/WHOIS,* RFC 954, October, 1985.

E. Krol, *The Hitchhikers Guide to the Internet,* RFC 1118, September 1989.

L. McLaughlin III, *Line Printer Daemon Protocol,* RFC 1179, August 1990

J. Postel, *Character Generator Protocol,* RFC 864, May, 1983.

———, *Daytime Protocol,* RFC 867, May, 1983.

———, *Discard Protocol,* RFC 863,May, 1983.

———, *Echo Protocol,* RFC 862, May, 1983.

J. Postel, K. Harrenstien, *Time Protocol,* RFC 868, May, 1983.

J. Postel, J. Reynolds, *File Transfer Protocol (FTP),* RFC 959, October 1985

———, *Telnet Protocol Specification,* RFC 854, May 1983.

Barry Shein, "The Telnet Protocol," *ConneXions,* v. 3, no. 10, October 1989, pp. 32-38.

K. R. Sollins, *The TFTP Protocol (Revision 2),* RFC 783, June, 1981

James VanBokkelen, "Workstation Oriented Enhancements to Telnet," *Connexions,* v. 3, no. 7, July 1989.

Chapter 15

管理

15.1 ネットワークの管理

ネットワーク管理システムには，3つの要素がある．

- マネージャ
- エージェント
- 被管理オブジェクト

被管理オブジェクトは，ネットワーク上のモジュールである．Unixのカーネル，ルーティングモジュール(routedなど)，ネームサーバ，自動販売機，その他のネットワークにつながっている面白いものは何でも構わない．実際，例えばイーサネットケーブルや一連のdumb装置のように，オブジェクトの定義は1つのマシンの境界からはみ出している．人間は，これらの対象のどんなものでも，観察し，操作しようとするものである．

マネージャとは，人間に代わって管理要求を発行するソフトウェアのことであり，監督者(director)の役割を果たす．このような要求の多くはプログラムにすることができる．例えば人間ならば監督者にあるデータを10分おきに欲しいと指示できる．

エージェントとは遠隔ノードにあるソフトウェアのことで，マネージャの要求に答えて，その要求を被管理オブジェクトの内部呼出しに変換することのできるものである．エージェントとマネージャとの間で使われている言語は，ネットワーク管理プロトコルと呼ばれており，単純ネットワーク管理プロトコル(Simple Network Management Protocol : SNMP)とRPCに基づいた管理サービスの2種類の言語を用いることができる．

被管理オブジェクトは，マネージャのことを知っていなくても構わない．エージェント/マネージャモデルの利点の1つは，1つのエージェントが，多くの異なるノード上の多くの異なるマネージャからの要求を処理できることである．このときエージェントは，実際のモジュールや被管理オブジェクトを，どう制御するかを明らかにすることに責任をもっていなくてはならない．

エージェントは，被管理オブジェクトに命令を発行することがあるかもしれないが，たいてい

のやりとりはもっと穏やかなものである．オブジェクトはいくつかの属性をもち，それらのうちのいくつかは，そのオブジェクトの状態を表している．例えば，あるイーサネットカードの状態を示す属性値として，送信されたバイト数が挙げられる．

しかし，他の属性の中には，そのオブジェクトの特性を決定するものもある．オブジェクトを直接制御するのではなく，これらの特性的な属性の1つを書き替えれば，オブジェクトはそれに気づいて動作を変更するだろう．

この章では，まず最初に管理されるものの定義から始める．管理情報ベース(management information base : MIB)の標準的な定義があれば，異なるベンダの製品でも，もしそれらが同じ基本的な機能をもっていれば，単一のマネージャ(もしくは異なるベンダからの複数のマネージャ)によって，すべて制御できることを意味する．

実際には管理されるものの定義には2つの部分がある．まず第1に，管理されるものに名前を付ける方法が必要である．この方法は，管理情報の構造(structure of management infromation : SMI)と呼ばれている．第2に，オブジェクト自身の定義があり，この方法は様々なMIB RFCの中で定義されている．TCP/IP階層(stack)の中心となるオブジェクトはMIB II(core MIBの第2版)で定義されている．他のMIBにはT1ポート，シリアルポート，さらにはトースターのような特化したデバイスが定義されている．

本章ではオブジェクトの定義を議論した後，それらのオブジェクトに対して要求をどのように伝えるかについて考察する．これは単純ネットワーク管理プロトコル(SNMP)の仕事である．SNMPはTCP/IPに基づいた，マネージャとエージェントとの間の通信のための標準的な言語である．

第3では，Sunのネットワークにおいて各種のマネージャを設定するためのプラットホームであるSunNetマネージャについて見る．SunNetマネージャはSNMPオブジェクトを管理するが，それに加えてRPC上に作られたオブジェクト管理のための言語であるSunNetマネージャコールを用いることもできる．

ネットワーク管理の問題に対する他の取組みにも触れておこう．例えばDECは，DECnet Phase Vの固有のインターフェイスとして，SNMP言語ではなくCMIP言語を定めている．また，DECはマネージャのための異なるアーキテクチャをもっていて，DECmccとして知られている．しかし，SNMPに基づく方式の方が標準的であり，広く利用可能なのに対して，CMIPの実装はベンダごとに少しずつ異なっている．

異なるアーキテクチャにまたがって管理を行うことは簡単な問題ではない．SunNetマネージャは，TCP/IPとDECnetの両方の環境を管理するのに利用することができるが(DECnet本来の管理プロトコルで通信するためには，NICEエージェントを用いる)，本来のプロトコルを使うことにはいつでも何らかの利点がある．1つの方法は，1つのネットワークでは単一のプロトコルだけを用いるようにすることである．もしこれが可能でないと，多くの環境では，いくつかの異なる

管理ツールを用いるはめになる．

マネージャ間の通信プロトコルを作成しようという試みがいくつか行われている．この場合，マネージャとは人間ではなくソフトウェアモジュールを指している．人間の間の対話を改善することについても述べるべきことがあるのだが，それは他の本で述べられるべきことである．

15.2 管理情報の構造

「管理情報の構造と識別(the structure and identification of management information)」とは，管理されるオブジェクトに名前をつける標準的な方法を表す面白い言葉である．RFC 1155はこの印象的な表題をもっているが，被管理オブジェクトを登録するための手続きを定義している．オブジェクトの数や属性は非常に多いので，標準的な方法を用いることが不可欠である．

名前づけには，2つの基本的な部分がある．1番目は，オブジェクトを識別するための方法である．これには，この本に述べられている他の名前づけサービスに似た階層的な名前の木を用いる．2番目は，これらのオブジェクトが何であるかを表すのに用いる，いくつかの標準的なデータ型である．

RFC 1155は，したがってメタスキーマである．つまりそれは，実際のオブジェクトは何も定義せずに，オブジェクトを定義する方法のみを定義している．オブジェクトそのものはMIBの中で定義される．オブジェクトにアクセスするには，SNMPの知識と下位のMIBに関する知識が少し必要である(ただし，後で見るように，すべてのオブジェクトについて知りつくしていなくても，“木の中を動き回る”ことは可能である)．

管理されるオブジェクトはすべて名前，文法，そして符号化(encoding)規則をもっている．名前は，オブジェクト識別子(object identifier)によって一意に表現される．オブジェクト識別子は，管理の都合上割り当てられた名前である．文法は，ASN.1の非常に制限されたサブセットである．符号化規則は，そのオブジェクト型のインスタンスがどのように表現されているかを表している．文法と符号化規則はネットワーク上の伝送と密接に関連していて，例えばSNMPは，ASN.1に付随している基本符号化規則(Basic Encoding Rules : BER)のサブセットを用いている．

15.3 名前

名前は本質的に階層的なものである．ここでは，それぞれのレベルにおいてオブジェクトに整数を割り当てる識別モデルが使われる．したがって，オブジェクトの識別子は，木全体を渡り歩く整数の列になる．人間にとって読みやすくするために，これらのオブジェクト識別子にはそれぞれ名前がつけられている．それぞれのオブジェクトが，その下に部分木をもってもよい．例えばデータベースという名前のオブジェクトがあると，その下には表とかユーザといったサブオブ

ジェクトがありそうである．サブオブジェクトがまた，さらにサブオブジェクトをもつこともできる．

木をどんどん低いレベルまで降りていけば，ドメイン名前システムで見たように，管理権限を委任することができるようになる．木の項上は Internet Assigned Numbers Authority が管理している．低いレベルは特定のベンダやユーザによって管理されているかも知れない．

木の根元にはラベルがつけられていないが，少なくとも ISO(1), CCITT(0), ISO/CCITT 共同(2)という 3 つの子供がある．したがって，1 から始まるオブジェクト識別子は ISO の勢力下にある．2 から始まるすべての番号は，競合する 2 つの組織の共同管理下に置かれている．

ISO の木の下には，機関を示すオブジェクト識別子 3 があり，その下には合衆国国防省(DoD)を示すラベル 6 がある．したがって，オブジェクト DoD を参照するためには，オブジェクト識別子 1.3.6 が用いられる．

DoD は，その木の一部を Internet に譲っている．その結果，オブジェクト "Internet" は次のように定義されている．

```
Internet OBJECT IDENTIFIER ::=
    {iso org(3) dod(6) 1}
```

Internet Activities Board は，この空間をさらに 4 つに分割した．

```
directory       internet 1
mgmt            internet 2
experimental    internet 3
private         internet 4
```

directory サブツリーは，OSI ディレクトリを Internet の中でどのように使うかを議論するための，将来の文書のために予約されている．mgmt は IAB が認可した標準の文書のために使われる．この部分木の下の管理は，Internet Assigned Numbers Authority に委任されている．

例えば，もともとの MIB は RFC として発行されていて，次の番号が割り当てられている．

```
1.3.6.1.2.1
```

Experimental も Assigned Numbers Authority によって管理されている．最後に，private がある(これも Assigned Numbers Authority により委任されている)．private 部分木には，事業者(enterprise)に委任されたノードがある．

```
enterprises OBJECT IDENTIFIER ::= {private 1}
```

enterprises には，ベンダが自社の製品のモデルを登録することができる(登録者は ISI の Joyce K. Reynolds で，電子メール jkrey@isi.edu で連絡をとることができる)．それぞれのベンダは，将来衝突が起きないように，自社の製品を登録することが推奨されている．

```
companya ::= {enterprises 2}
product1 ::= {companya 3}
```

ここまでくると，オブジェクト識別子を割り当てることができる．オブジェクトに名前をつけることに加えて，そのオブジェクトの属性(木の下の方で識別番号が与えられる)も参照できるようにしたい．

基本的なデータ型の基本集合が定義されていて，ほとんどのMIBの定義で利用することができる．データ型をかなり簡単なものに抑えたのは，ほとんどのマネージャが，ほとんどの型のオブジェクトを見ることができるべきであるという考えに基づいている．基本的なデータ型としては，整数，オクテット列，オブジェクト識別子，ヌルなどいくつかのものがある．他にも，SMI文書で定義された6つの型のように，定義されたデータ型がいくつかある(図15-1参照)．数え上げの整数には，値0は予約済で利用してはいけないという非公式な習慣がある．これ以外に2つの構成的データ型がある．リストは基本的な型の列として表現され，表はリストの列である．

データが実際にネットワーク上を送信されるときには，(XDRとは異なり)ASN.1の基本符号化規則(Basic Encoding Rules : BER)のサブセットが使用される．ASN.1基本符号化規則は，自己記述的であるという特徴をもっている．つまり，ノードはデータそのものから，どんな種類のデータを受け取っているのかを知ることができるのである．

あるオブジェクトを定義するためには，5種類の情報が必要である．

・テキストでの名前(オブジェクト記述子)とオブジェクト識別子
・文法
・定義
・アクセス
・状態

6番目の要素であるインデックスは，複数のインスタンスをもつ変数において使用される．アクセスはオブジェクトがマネージャやエージェントから利用可能であるかどうかを示している．標準的なアクセスクラスはread-only, read-write, write-only, アクセス不可の4つである．ただし，MIBの中のアクセスの定義と，SunNetマネージャのオブジェクトによって提供されるアクセス制御とは異なっている．

オブジェクトのステータスは，インターネット標準としての状態を示している．必須(manda-

NetworkAdress	選択(現在は Internet 族のみ)
IpAdress	長さ4のオクテット列
Counter	32-bit 符号なし，単調に増加する整数(巡回するかもしれない)
Gauge	32-bit 増加/減少する符号なし整数で，最大値で停止する
TimeTicks	1/100秒単位で時間を計測する
Opaque	オクテット列に符号化された任意の値で，二重に包まれている

15-1 SMIで定義されているデータ型

tory)，省略可能(optional)，不許可(deprecated：推奨できないが，未だに残っている状態)，旧式(obsolete)の4つの状態がある．“必須”は，もし一般的なクラス(例えばTCPの実装)にあてはまる製品があるならば，そのオブジェクトは必ず用意されなければならないことを意味している．

“旧式”は，かなり役に立つ状態である．というのは，オブジェクトを消去するのでなく，単に“旧式”と印をつけることで済ますことができる．旧式のオブジェクトと同じ名前をもつ新しいオブジェクトは存在することはできないというのが規則である．これにより，あるオブジェクト識別子を見れば，それがどんなオブジェクトを意味しているかわかるのである．

オブジェクトの型(MIBで定義される)と，オブジェクトのインスタンスは異なるものである．オブジェクトのインスタンスは，オブジェクトの型が実体化(instantiation)されて値と結びつけられたものである．オブジェクトのインスタンスを参照するための方法は，簡潔なMIB形式のインデックスの一部分である．SNMPのようなプロトコルは，単純なオブジェクト型にアクセスするための手段を提供している．これ以外にも，大量データの取出しや木の散策といった集合的な操作を実行するための簡単な道具が用意されている．SNMPでは，これらはget-nextオペレータによって取り扱われる．

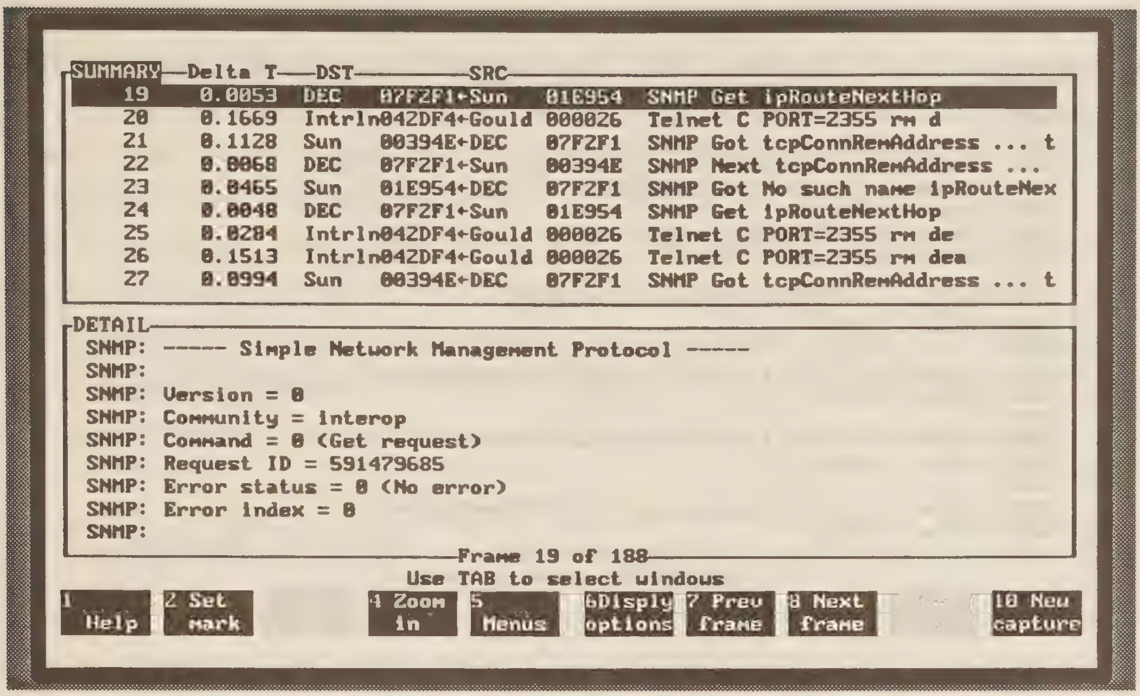
オブジェクト識別子は，バージョン番号によって識別される．新しいバージョンは，古いオブジェクト型を旧式だと宣言することができるが，それらの名前を削除することはできない．また新しいバージョンは，集合的でない他のオブジェクト型を，オブジェクト型の定義リストの中に加えたり，新しいオブジェクトの型を定義したりすることで，オブジェクト型の定義を補強することができる．しかし，名前を変えない限り，以前に定義されたオブジェクトの意味(semantics)を変更することはできない．

15.4 SNMP

単純ネットワーク管理プロトコル(Simple Network Management Protocol：SNMP)は，その名前からわかる通り，2つのノードが管理情報を通信するための簡単な方法である．このプロトコルについての深い議論は，SNMPを定義しているRFCや，Marshall Rose著“*The Simple Book*”で述べられている．

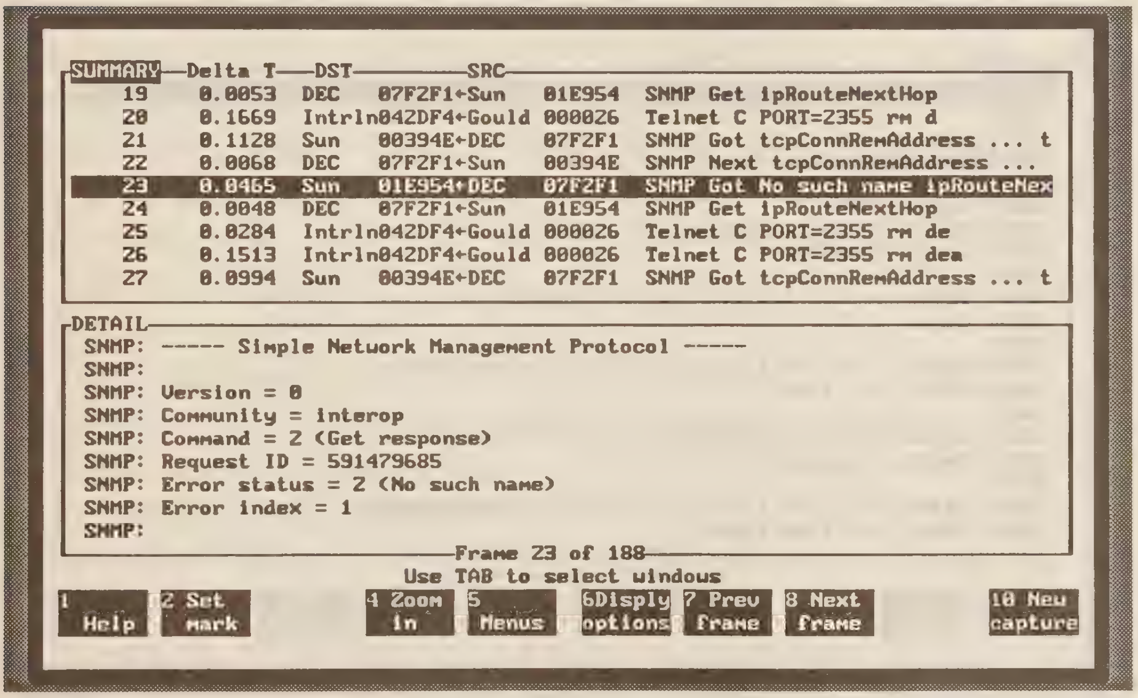
ここではSNMPの操作をネットワークから得られた4つの画面で見ることにしよう(図15-2から15-5を参照)．基本的な操作は図15-2に示されているget要求である．要求は要求IDによって識別され，返ってくる答と対応づけることができるようになっている．

このパケットは外に出ていく要求なので，エラーフラグもエラーインデックス値もゼロに設定されている．communityフィールドはInteropになっている．これは，San Joseで毎年行われているINTEROPの展示会において，このパケットが捕まえられたことを示している．図15-2の中でこれに続くパケットは，データが返されて，get-nextメッセージが発行されたことを示してい



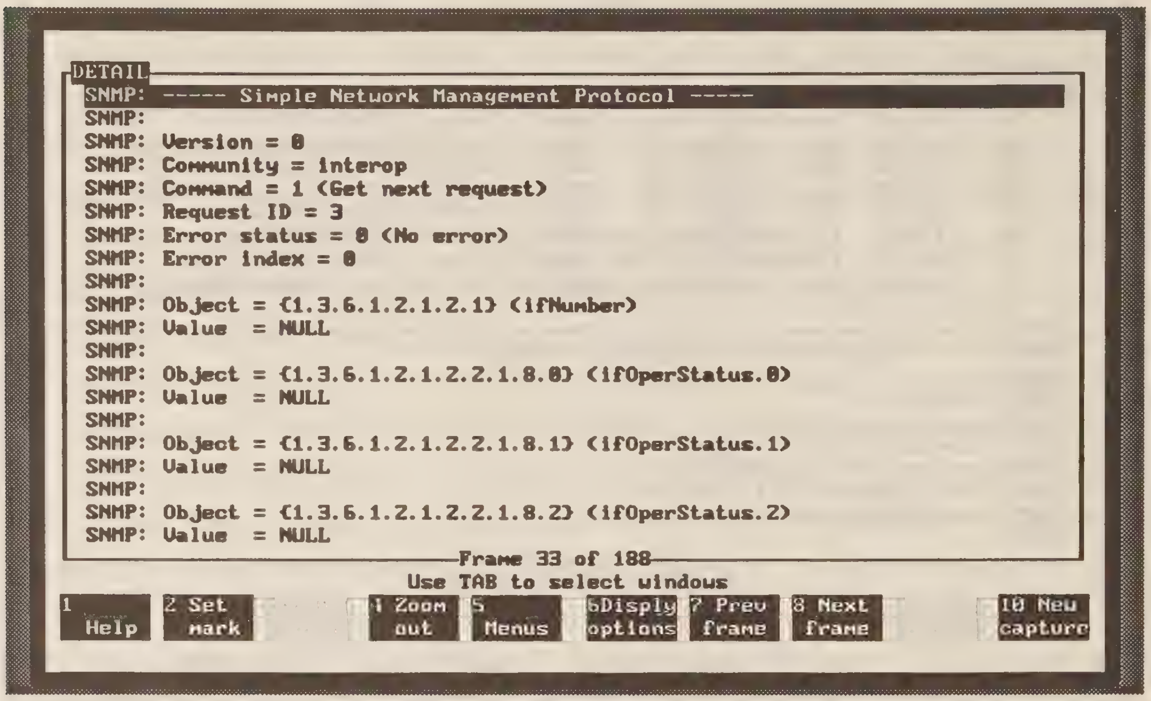
Courtesy of Network General

15-2 get要求



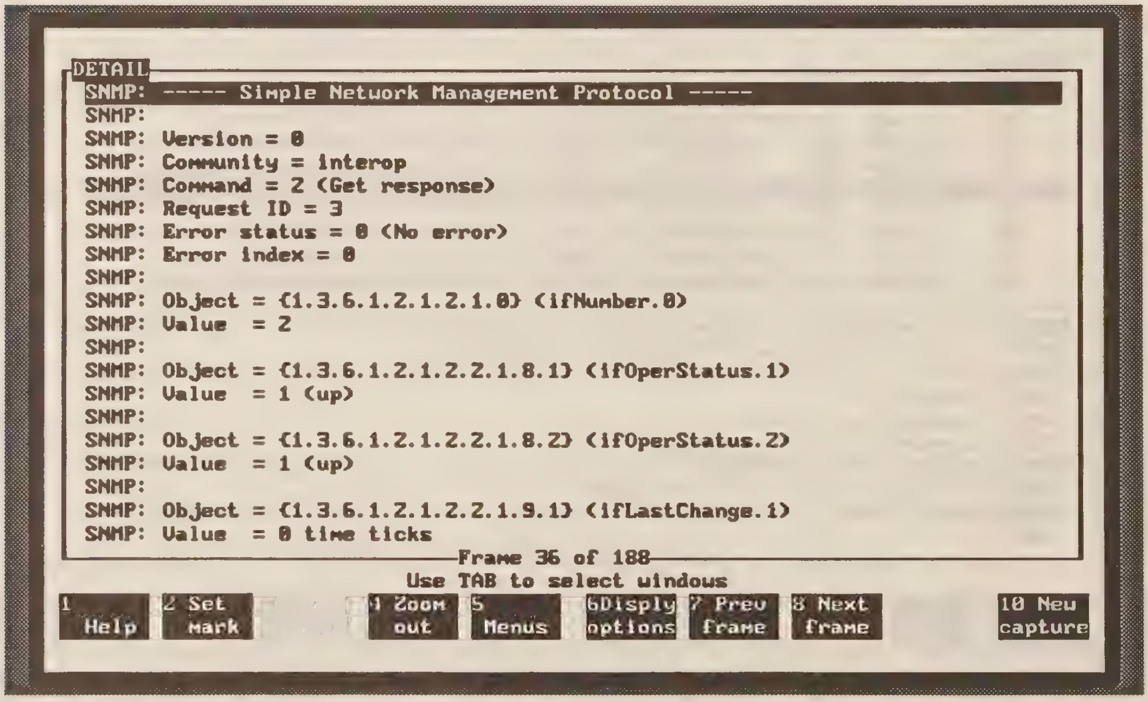
Courtesy of Network General

15-3 誤り応答



Courtesy of Network General

15-4　get-next要求



Courtesy of Network General

15-5　get-next要求に対する応答

る．get-next コマンドに関しては，すぐ後で説明する．

図 15-3 は前回の要求に対する解答を示している．要求 ID が前の図と一致していることに注意してほしい．エラーは“no such name”エラーであり，問い合わせられた装置には，その識別番号をもつ変数はないことを意味している(名前は，SMI に基づいたオブジェクト ID の単なる略記にすぎないことを思い出してほしい)．

図 15-4 は get-next コマンドを示している．オブジェクト ID が与えられて，オブジェクト ID 階層内での位置に基づいて，次のオブジェクト ID に対して問合せが行われる．このコマンドは，いくつかの異なるオブジェクト，典型的には 1 つの表のすべての列などを指定することができる．したがって，get-next は表のすべての行を見るのに使用することができる．図 15-5 は，前回の要求での変数のそれぞれが，新しい変数とその値をどのように返してきたかを示している．

15.5 SunNet マネージャ

SunNet マネージャは，開発プラットフォームであると同時に，管理システムでもある．一連のモジュールがソフトウェアに付属してくるので，様々な異なる種類のエージェントを通して管理対象のオブジェクトをアクセスできる．標準的なソフトウェアモジュールを用いれば，ネットワーク上のノードを見い出したり，SNMP による管理，定期的なデータレポートのグラフ化，その他様々なことを行うことができる．

さらに，このソフトウェアには一連のライブラリや定義が含まれていて，ユーザやソフトウェア開発者が他の種類のモジュールを追加することができるようになっている．新しい MIB を組み込むこともできるし，他のエージェントと通信するのに NICE のような言語を利用することもできるし，また情報を処理するために表示モジュールや機能モジュールを追加することもできる．

開発者の視点から見たとき，このソフトウェアは 3 つの基本的な API をもっている．

・マネージャサービス API

・エージェントサービス API

・OpenWindows API

OpenWindows はユーザインターフェイスにアクセスするためのライブラリである．Display PostScript と X-ウィンドウの組合せに基づいているので，プログラマはこのライブラリを使って画面上に情報を表示することができる．

マネージャサービス API も，アプリケーション開発者のためのインターフェイスである．これによって，プログラマは SunNet マネージャ環境にアクセスすることができ，ソフトウェアからネットワーク上のオブジェクトを見つけたり質問したりすることができる．新しいエージェントを追加するには，エージェントサービス API を利用する．エージェントサービス API を利用することによって，エージェント開発者は RPC 呼出しの構成方法や，マネージャとの通信の仕方や，そ

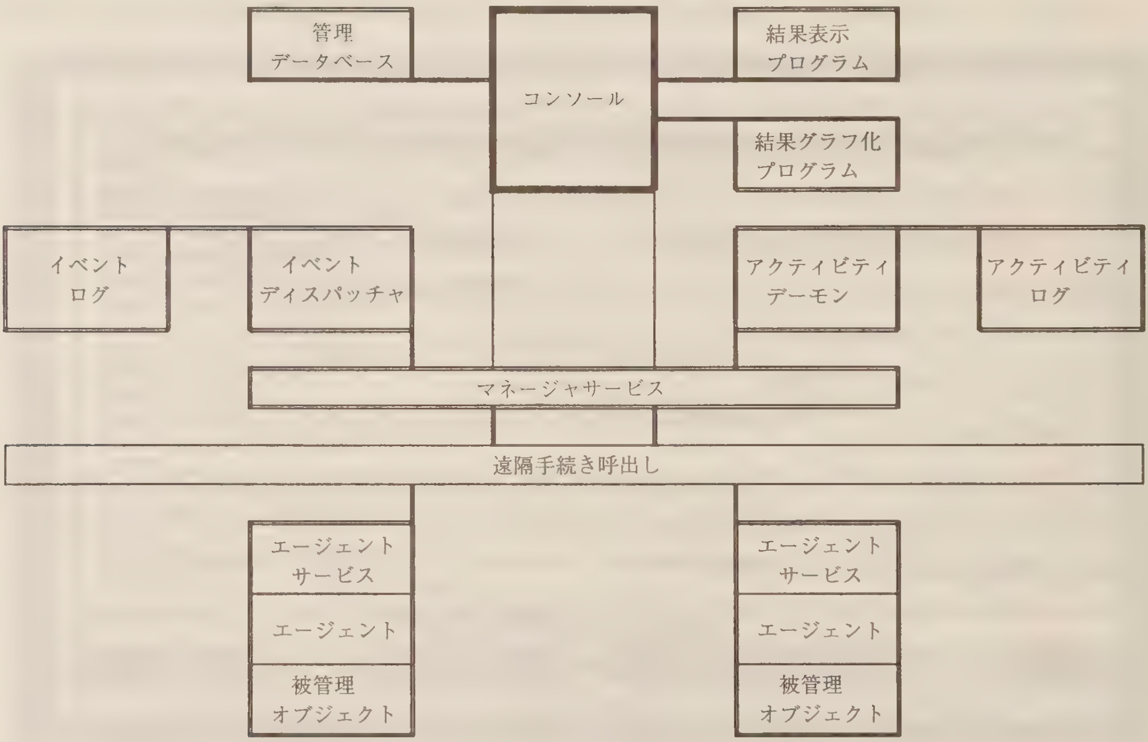
の他の管理上の仕事の細部を知らなくて済むようになる．

SunNet マネージャコンソール(console)は，OpenWindows をもったワークステーションで動作する．ワークステーションはこの仕事に専念しなくてもよいが，たくさんの要求があればたくさんのデータがやってくるので，かなり急速に資源を消費する．

コンソールのユーザインターフェイスを使って，2 組の基本的なサービスを利用することができる．表示サービスはユーザの画面に対応している．それに加えて選択サービスがあり，マネージャは特定のネットワーク構成要素を見つけて，希望する機能を起動することができる．選択サービスを利用すれば，マネージャはネットワークを構成要素の階層として組織的にとらえることができる．選択および表示サービスの下には要求管理サービスがある．これは下位のマネージャの変化をコンソールから切り離している．

図 15-6 は SunNet マネージャアーキテクチャの簡略図である．コンソールはユーザが作業をするところである．コンソールは，マネージャサービスライブラリを利用して，ある型のデータに対する要求を発行する．マネージャサービスライブラリは RPC プロトコルを使ってネットワーク上の様々なエージェントとその情報をやりとりする．

代理エージェント(proxy agent)と呼ばれるいくつかのエージェントは，SNMP の代理エージェ



15-6 SunNetマネージャの構成要素

ントの場合のように，管理しているエージェントにデータを転送する．他のエージェントは被管理オブジェクトに代わって直接応答し，その結果を送り返す．

いったん SunNet マネージャのコンソールに情報が受信されると，結果グラフ化プログラムやその他特定の環境のために作成されたモジュールによって，その情報を表示させることができる．標準の API を用いれば，新しいエージェントや管理モジュールを追加するのは比較的簡単なことである．

15.5.1 エージェント

エージェントライブラリには，すべてのエージェントで必要とされる基本的な機能を備えていて，エージェントがイベントやエラー，監視情報を送る機能を提供している．また，このライブラリを利用して，エージェントは様々なマネージャとの接続を初期化することもできる．

様々なエージェントが，基本的な SunNet マネージャソフトウェアに標準的に付随している．図15-7 にはそれらの基本的なエージェントのうちのいくつかを挙げた．さらに，新しいエージェントを付け加えるのは，ほとんど自明の作業である．開発者用のツールキットには，エージェントのソースコードの見本や指導書(tutorial)，そして新しいエージェントが正しく動作しているかどうかを確認するためのテストマネージャが付属している．

この環境におけるほとんどのエージェントは，RPC および XDR プロトコルを用いて SunNet マネージャのコンソールと通信している．このようなインターフェイスをとっているので，secure RPC 認証によって，エージェントが正しいマネージャだけから管理されていることを確認するこ

SNMP	SNMP代理エージェント
PING	ICMP echoによる到達可能性，パケットの往復時間のテスト
HOSTIF	一般的なインターフェイスの情報(例えば，送信・受信・エラー・衝突したパケットの数)
SYNC	同期インターフェイスエージェント
X25	仮想回路の情報，現在のリンク状態の情報
ETHERIF	イーサネットインターフェイス
LAYERS	TCP/IPプロトコル集に関する情報
IPROUTES	経路選択テーブルの情報
TRAFFIC	LANの通信量
RPCNFS	RPCとNFSの統計情報
HOSTPERF	rstat ONCサービスを利用する
HOSTMEM	ネットワーク構造体に使用されているホストのメモリ．Sunでは，STREAMSとmbufに関する情報
DISKINFO	ディスクの利用情報
IPPATH	IPノード間の接続可能性の情報
LPSTAT	プリンタの状態情報

15-7 SunNetマネージャのエージェント

とができる．RPC の下では UDP 転送プロトコルが用いられる．

いくつかのエージェントは代理エージェントであり，SunNet マネージャ手続きライブラリの中の標準的な RPC 呼出しを受け付ける．その後，代理エージェントは要求を変換し，他の何らかの手続きを通して送り出す．例えば，多くのオブジェクトは SNMP を用いて管理できる．本来の SNMP オブジェクトをアクセスするのに SNMP 代理エージェントを用いることができる．

代理エージェントはまた，有用なフィルタ機能を実行することもできる．エージェントに対するよくある要求は，ある活動，例えばホストのクラッシュが起こったときに知らせてほしいというものである．ホストがクラッシュしているかどうかを知るためには，誰かが定期的にすべてのホストの状態を調べて，動いているかどうかを確かめなくてはならない．

もしその状態確認機能が広域ネットワーク上をまたいで実行されれば，高価なバンド幅がすぐに使い尽くされてしまうだろう．これに対して，代理エージェントがこの機能を実行すれば，広域パケットは興味の対象となるイベントが起こったときにだけ必要となる．

15.6 管理データベース

SunNet マネージャコンソールは管理データベースを使用しているが，それはネットワークを記述するスキーマファイルやインスタンスのファイルを集めたものである．スキーマファイル(*.schema)はオブジェクトのデータ構造を格納している．オブジェクトのインスタンス(およびそれらと他のオブジェクトとの関係)はインスタンスファイル(*.db)に蓄えられている．

分散環境では，もう 1 つの RPC に基づくプログラムである NFS を利用することができる．スキーマファイルとデータベースファイルを遠隔サーバに格納して，複数のマネージャによって共有してはいけない理由はどこにもない．1 つの特定の操作だけに必要なファイルは，局所的にマウントしたり，アクセスの制限された NFS マウントにしてもよい．SunNet マネージャをデータベース管理ソフトウェアと組み合わせて利用することも可能である．

15.6.1 SunNet マネージャの各要素

管理できる要素の型は拡張することができる．SunNet マネージャは element.schema というファイルをもっていて，それには要素の 4 つの基本型があらかじめ組み込まれている．要素のカテゴリーは固定されているが，カテゴリーに属する要素は簡単に増やすことができる．要素の 4 つのカテゴリーは次の通りである．

・構成物(例：プリンタ，ルータ，ワークステーション)
・ビュー(例：他のビューを含む要素の集まり)
・バス(例：イーサネット LAN セグメント)
・接続(イーサネット，専用回線，RS-232 リンクのように 2 つの要素を接続する要素)

それぞれの要素の型は，レコードを使ってスキーマの中で定義される．例えば，Sun-4，サブネットワーク，トースターといった要素が定義できる．どんな構成物やビューに対しても，レコードインスタンスの型を利用してグリフ（glyph：要素を表示するために使われるアイコン）を指定することができる．

余談だが，Epilogue Technology 社の John Romkey の努力のおかげで，トースターは SNMP の標準的なオブジェクトとして利用されている．INTEROP 展示会のネットワーク設計者の1人である Romkey は，どんなものでも，たとえトースターでもネットワークにつなぐことができると一度冗談をいったことがある．Interop Inc. の社長である Day Lynch は彼をけしかけ，その結果，最初のインターネットトースターができ上がった．トースターを ON にしたり，PopTart を ON にする MIB にアクセスするのに SNMP が使用されている．

ある型の要素に対して適用されるユーザ命令を定義することさえもできる．ユーザがグリフを指すと，その要素に適用可能なコマンドを表示するメニューをプルダウンすることができる．rlogin と telnet の2つのコマンドはデフォルトだが，他のものを加えることもできる（図 15-8）．

```
record component.sun4 (
            string(32)      Name
            string(40)      IP_Address
            string(40)      User
            string(40)      Location
            string(80)      Description
            string(60)      _hidden_field
      )
record view.subnet (
            string(32)      Name
            string(80)      Description
      )
record component.toaster (
            string(32)      Name
            string(80)      Description
            string(20)      BrandName
      )


instance elementGlyph (
      ( component.sun3            sun3.icon)
      ( component.sun4            sun4.icon)
      ( component.toaster         toaster.icon)
      )

instance elementCommand (
      (component.sun4      ``MyUtility''      ``myutility %Name'')
      )
```

15-8　要素の定義

```
cluster (
        component.sun4 (
                SunEng14
                1.1.2.288 Engineer23
                "Building 2, Room 146"
                "Sun-4/110 - monochrome, diskfull" )
        membership ( Subnet2)
        agent ( etherif )
        agent ( hostif )
        proxy ( hostperf SunEngServer )
        agent ( layers )
        proxy ( ping SunEngServer )
        agent ( rpcnfs)
        glyphColor ( 220 220 250 )
        connect ( SunLAN)
)
```

15-9 構成要素の定義

ある要素の実際のインスタンスも，クラスタレコードとして，テキスト形式でデータベースに格納されている．クラスタレコードは複数のレコードの集合体で，それで 1 つのインスタンスをなすものである．例えば図 15-9 は，ある特定の Sun-4 の要素に適用されるクラスタレコードを示している．

このクラスタレコードによると，その Sun-4 はある特定のビュー要素の一員であり，特定のバスに接続されている．さらに，その要素で利用可能なエージェントや，代理として動作できるもう 1 つのサーバの名前が記述されている．代理というのは，SunEngServer 上で動作している代理のエージェントが，情報を送り返す責任を担っているので，もしこの Sun-4 に質問をしたければ，要求はこの代理エージェントに対して送らなくてはならないという意味である．

15.6.2 データレコード

処理中の要求(active requests)もまたデータベースのレコードとして表されている．データ要求レコードは，返されるべきデータや，それがどう表示されるか，その要求を表示するグリフをどこに置くかを定義している．

興味の対象となるイベントは，マネージャごとに異なるものである．あるマネージャにとっては，ユーザが多くの資源を使い過ぎている状況に興味があるだろう．他のマネージャはセキュリティ違反に興味があるかもしれないし，さらに別のマネージャはシステムの性能に興味をもっているかも知れない．これらの異なる種類の要求を処理するために，SunNet マネージャアーキテクチャではイベントとアクティビティ(activity)という概念を用いている．イベントとは，監視していた特性が変化したというエージェントからの報告である．ユーザは，自分がある特定の種類のイベントの報告に興味があると宣言しておくことができる．アクティビティとは繰り返し起動さ

れる監視機能である．同様にユーザは，ある種類のアクティビティの報告を維持すべきであると指定することができる．

それぞれのコンソールにはディスパッチャサービスがあり，エージェントから報告されたイベントの待ち合わせ場所の役割を果たしている．ディスパッチャはイベントの報告をユーザに渡したり，レポートをイベントログにすることもできる．イベントログには，被管理エージェントからの情報だけが含まれているとは限らず，マネージャ上のプログラムもまたログに情報を残すことができる．例えば，SunNet マネージャ自身も，エラーを報告するのにこのログを利用している．

イベント報告の特殊な形式として，警告がある．ある属性が指定したように変化したときに，ディスパッチャがアイコンの色を替えるといったような特定の動作を行うように，ユーザが要求を出すことができる．例えば，あるユーザはセキュリティ侵犯がなされたかどうかの警告に興味があるだろうし，他のあるユーザは CPU の利用度がある点に達したことに興味があるだろう．

アクティビティはアクティビティデーモンによって処理される．エージェントが最初のアクティビティ要求に答えたときに，このアクティビティがアクティビティデーモンとともにログに蓄えられる．デーモンは要求された動作がまだ行われていることを 30 分ごとに確認する．もし動作していなければ，そのアクティビティはアクティビティログから削除される．ユーザはアクティビティログを見て，どの要求が活動中であるかを調べることができる．必要ならば，失敗した後で要求が自動的に再起動されるようにすることもできる．

15.6.3 管理データベース中でのアクティビティ要求

アクティビティとデータの要求はすべて管理データベースに保存されていて，一連の標準的な要求を異なるユーザや異なるコンソールシステムの間で共有できるようになっている．図 15-10 から 15-12 は，そのような定義の例を示している．

図 15-10 はデータ要求レコードを定義しているクラスタの始まりを示している．この要求は名前をもっていて，アイコンとして画面上に表示することもできる．イベントが実際にどのように画面上に表示されるかを決めるのは，定義クラスタの 2 番目の部分のデータ属性レコードである（図 15-11 参照）．

最後に，図 15-12 はイベント要求を示している．このイベント要求によってマネージャは，ある種類のシステムからの，ある種類のイベント報告に興味があることを自分で登録できるようになっている．会計係は「口座の引出し超過」イベントを登録するであろうし，ネットワーク管理プログラムは「不正なパケット」，CEO は「エンジニアリング部門で誰もログオンしていない」を登録するであろう．

cluster(	あらゆる要求は，活動中のものも停止中のものもクラスタレコードで定義される．
dataRequest(	データレポートの始まり: データ要求レコード
hostperf.data.0	要求の名前．要求のグリフの下や要求に関するメッセージの中に表示される．
0	要求の通し番号．0は新しい要求を示す．
SunEng14	目標システムの名前
SunEngServer	代理システムの名前
hostperf	エージェントの名前
data	グループや表の名前
" 'Interval' 'Count' Times"	報告時間．一度か，intervalごとにcount回か， intervalごとにキャンセルされるまで．
10	エージェントに渡される時間間隔フィールド
5	エージェントに渡される回数フィールド
" "	エージェントに渡されるオプション
" "	エージェントに渡されるキー
" "	データ報告のためのファイル名．もしnullならば，報告はデータログフィールドに送られる．
undefined	コンソールの"fileport"内部変数．新しい要求ではundefinedに設定される．
True	要求が死んでしまったときに自動的に再起動すべきかどうかを示すブール値．
False	コンソールから要求があるまで応答を遅らせるべきかどうかを示すブール値．
"Save Request"	delete requestまたはsave requestの値をとり，要求が終了したときの扱いを示す．
undefined	コンソールのための内部タイムスタンプ．新しい要求の場合は undefined に設定される．
hostperf.data	エージェントとグループ名(上記のエージェントおよびグループ/表変数と一致しなくてはならない)．
SunEng14)	要求に対応した目標オブジェクト. データベースの中の要素の名前で，エラーメッセージに書き込まれる．

15-10 データ要求レコード

15.7 発見ツール

管理データベースを作成するには，すでにあるものから始めてもよいが，発見ツールを利用することもできる．このツールはバックグラウンドで動作し，自動的にネットワークの図形的な表現を作成してくれる．

発見ツールがホストやネットワークを発見した後で，ユーザは図形エディタを使用して図形表現に変更を加えることができる．図形エディタの中から直接新しいオブジェクトを作成することもできる．

発見ツールは特権ユーザモードで実行される．ユーザがこの機能を実行すると，ポップアップ

dataAttribute (	0個以上のデータ属性レコードは，データがコンソールにどのように表示されるかを制御する．
cpu%	属性の名前
True	データがデータログに表示されるべきかどうかを示すブール値．
1	ASCIIポートを示す内部コンソール変数
True	属性がグラフとして表示されるべきかどうかを示すブール値．
1473696	棒グラフポートを示すコンソールの内部変数
None	属性が折れ線グラフで表示されるべきかどうかを示す．値はnone，絶対値，差分値のいずれかである．
0	折れ線グラフポートを示す内部変数
"Delta Values"	属性がグラフでどう表示されるかを示す．値は，none，絶対値，差分値のいずれかである．
1515136	グラフプログラムのハンドラ(内部コンソール変数)．
)	
rqstState(	要求状態レコード
Idle	要求の状態．activeまたはidle．
0	内部変数で常に0にセットされる．
0	内部変数で常に0にセットされる．
)	
membership(	データ要求の最後のレコード．どのサブビューにオブジェクトのグリフが現れるかを示す．
SunEng14	オブジェクトの名前．通常データ要求レコードの最後の値と同じである．
)	membershipレコードの終わり．
)	data要求クラスタの終わり．

15-11　データ属性レコード

ウィンドウが現れて，rootのパスワードをたずねてくる．発見ツールはメインウィンドウに雲を描き始めるが，これらの雲はネットワークを表している．

ユーザは雲のグリフをクリックすることができて，そうするとそのネットワーク上のホストが表示される．ホストをバス(イーサネット)に接続したり，物理的なトポロジーに対応して画面上で再配置することもできる．

発見ツールの動作の仕方は，まず最初にルーティング表を見て他のノードを見つけることから始まる．同時に，外部から到着するネットワークトラフィックを眺めていて，他のノードについての情報を見つけ出す．このツールでは10ホップまで遠くを見ることにしている(つまり，大きなネットワークの中では非常に大量の情報がマネージャのデータベースに集められる)．いったんホストが発見されると，発見ツールはSNMPエージェントやSunが供給しているその他のエージェントを探す．

しかし，どんな発見ツールをもってしても，自動的に発見できない情報はある．例えば，誰が特定のノードを所有しているのか，誰が管理の責任をもっているのか，その他オブジェクトの物理的な場所などの情報を知りたい場合があるかもしれない．

cluster(	
eventRequest(	
hostperf.data.1	要求の名前
1	要求の通し番号
SunEng14	目標システム
SunEngServer	代理システム
hostperf	エージェントの名前
data	グループまたは表の名前
Once	報告頻度
0	時間間隔フィールド
1	回数フィールド
" "	オプションフィールド
" "	キーフィールド
False	再起動を示すブール値
False	最初の報告の後で要求を終了すべきか
False	要求されるまで応答を待つべきか
"Delete Request"	完了時の動作
undefined	内部タイムスタンプ変数
hostperf.data	エージェントとグループのID
SunEng14)	目標オブジェクト
eventAttribute	
cpu%	属性の名前
"Greater Than"	最初の閾値の大小関係
80	最初の閾値
"Threshold Not Set"	2番目の閾値の大小関係
" "	2番目の閾値
"Blink Glyph"	シグナル・オプション
" "	シグナルのファイル名かユーザ名
Low)	優先度(low, medium, または high)
rqstState(	
Idle	要求の状態
0	内部タイムアウト値
0)	内部要求フラグ
glyphState(	
32)	グリフの状態
membership(	
SunEng14	要求が現れるべきサブビューの名前
))	

15-12　イベント要求クラスタ

15.7.1　SNMP のサポート

Sun 上では SNMP は代理エージェントの形式で提供されていて，ルータなどの遠隔の装置を管理できるようになっている．Sun ワークステーション自身は他のエージェントを使って管理される．SNMP 代理エージェント(na.snmp)は SunNet マネージャコンソールと通信するのに RPC を使用している．代理エージェントは，そこから他のエージェントと通信するのに SNMP を使用している．

マネージャは標準的な SNMP MIB オブジェクトも使用できるし，事業者固有のオブジェクトを使用することもできる．代理エージェントが装置上の特定の SNMP エージェントへの要求を受け取ったときには，/var/adm/snm/snmp.hosts ファイルを探して，その装置のための SNMP スキーマファイルを見つける．もしその装置用の項目が見つからなければ，デフォルトのスキーマが使用される．

標準のデフォルトのスキーマはMIB II(あるいはMIB I)である．/etc/snm.confの中のna. snmp. default-schema というキーワードを見てみると，デフォルトのスキーマの場所がわかる．組織は専用の SNMP スキーマを作成してもいいし，ベンダのものを利用してもいい．

MIB を SunNet マネージャスキーマに変換する mib2schema というスクリプトがある．mib2schema を使ったあとで，記述テキストを改善したり，特性フィールドにフォーマット情報を追加したり，もし必要ならば文字列型の配列の大きさを変更したりすることもできる．

na.snmp-trap と呼ばれる非同期または予期しない報告を扱う特殊な代理エージェントがある．デフォルトでは，この代理エージェントは，トラップ報告をそのマシンのイベントディスパッチャに送る．複数の待ち合わせ場所に対してトラップ報告を送るように，代理エージェントに指示することもできる．

SNMP トラップファイルには，事業者固有のトラップが書かれている．そのファイルは一連のリストから構成されている．リストは(enterprise というキーワードで始まる)ID で始まり，その後オブジェクト識別子が続く．オブジェクト識別子の後には，トラップ番号と名前が続く．

```
# Sun Microsystems
enterprise   1.3.6.1.4.1.42
0            sun-specific-0
1            sun-specific-1
```

15.7.2　ネットワーク管理のセキュリティ

エージェントのサービスに対するアクセスを制限するためのセキュリティ機構を付加することができる．標準のままの状態では，どのエージェントから情報を質問してもよい．それぞれのエージェントに対して，マネージャはデータおよびイベント報告要求に対する読出しセキュリティ，

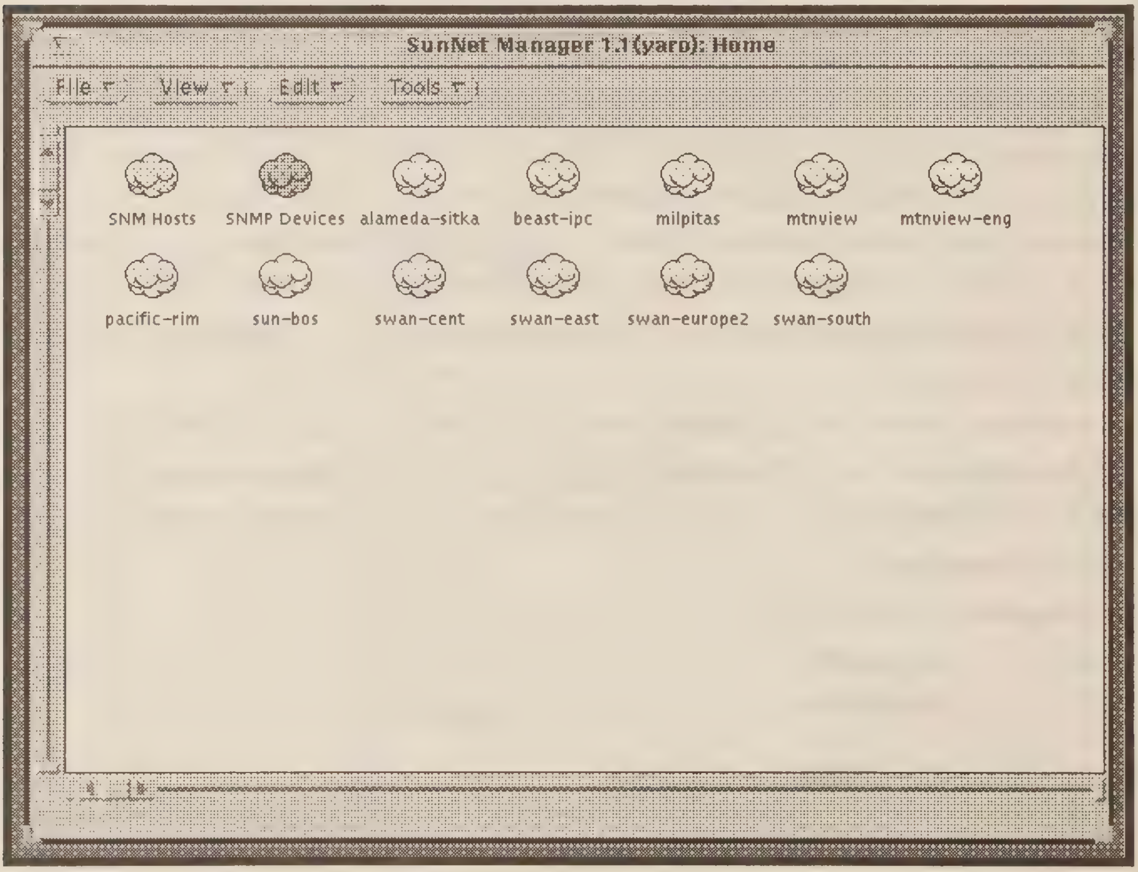
および設定要求に対する書込みセキュリティを指定することができる．

セキュリティは Secure RPC の上に作成されている．ユーザが誰であるかがわかった時点で，SunNet マネージャは，ネットグループに基づいてアクセス権をチェックする．アクセス権は，人々を 5 つの異なるネットワークセキュリティグループに分けることで管理される．

エージェントライブラリには，ネットワーク管理セキュリティレベルという大域変数があり，0（セキュリティなし）から最高セキュリティの 5 までの値をとる．この値の設定によって，どのクラスのユーザがエージェントを利用できるかが決められる．

セキュリティが有効になっているときには，エージェントは要求を許可するかどうかを決定するのに 6 つのステップを通る．まず，もし NIS が走っていなければ，その要求は却下される．次に，エージェントは自分自身のセキュリティレベルを参照する．もしレベルが 0 ならば，すべての要求が許可される．

次に，エージェントは，ユーザが Secure RPC 認証状をもっているかどうかを確かめる．もしそうであるなら，Secure RPC 認証状によって供給されるネットワーク名を，領域に固有のユーザ ID



Courtesy of Sun Microsystems

15-13 SunNetマネージャにおけるビュー

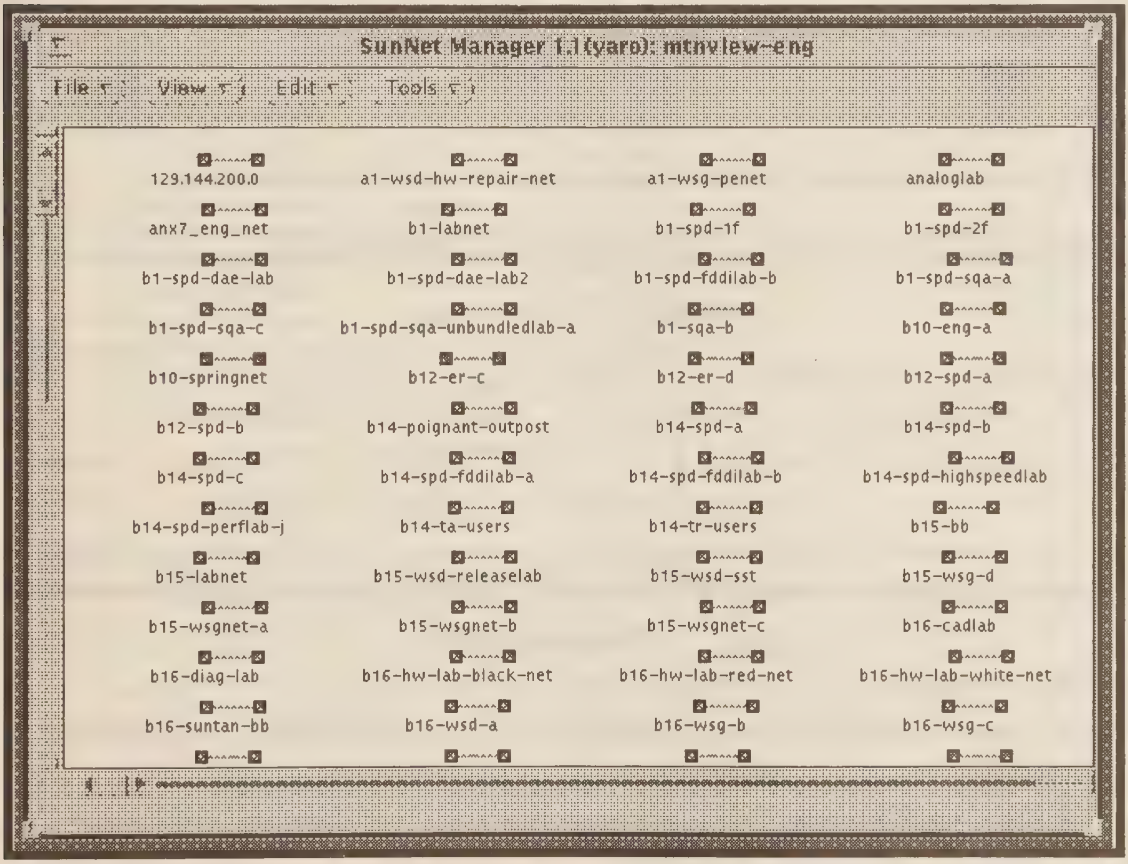
およびパスワード表の中から調べ出す．メンバーであることがいったん確認されれば，エージェントはセキュリティレベルを調べ，要求を許可するかどうかを決定する．

15.8 SunNet マネージャの動作

ネットワーク中のオブジェクトは，コンソール上では雲で表されるビューにまとめられる．これらの雲は，さらに大きな雲の一部であり得る．1つのネットワークは，最終的にトップレベルの地図に帰着するが，普通はこれは一連の単純な雲である（図 15-13 参照）．

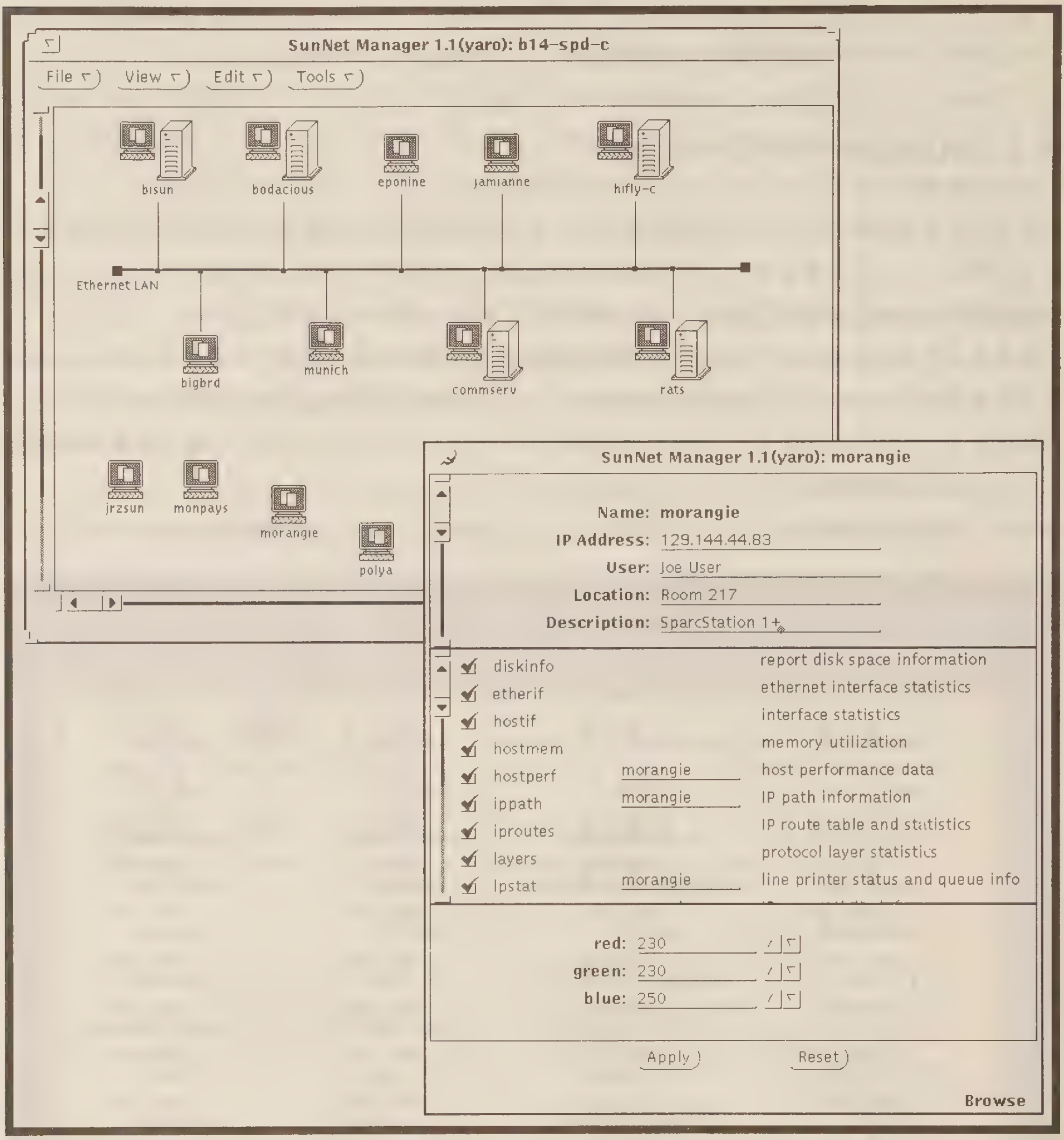
あるネットワークのトップレベルの地図を選択するのが基本的な操作である．あるビューをクリックすれば，そのビューの内容が表示される．これを何度か繰り返すと，実際のオブジェクトに到達する．Sun ワークステーション，その他のコンピュータ，それにもちろんシステム間の結合といったものが典型的なオブジェクトである．

すべての情報は管理データベースに蓄えられる．管理データベースに情報を収めるのは，テキ



Courtesy of Sun Microsystems

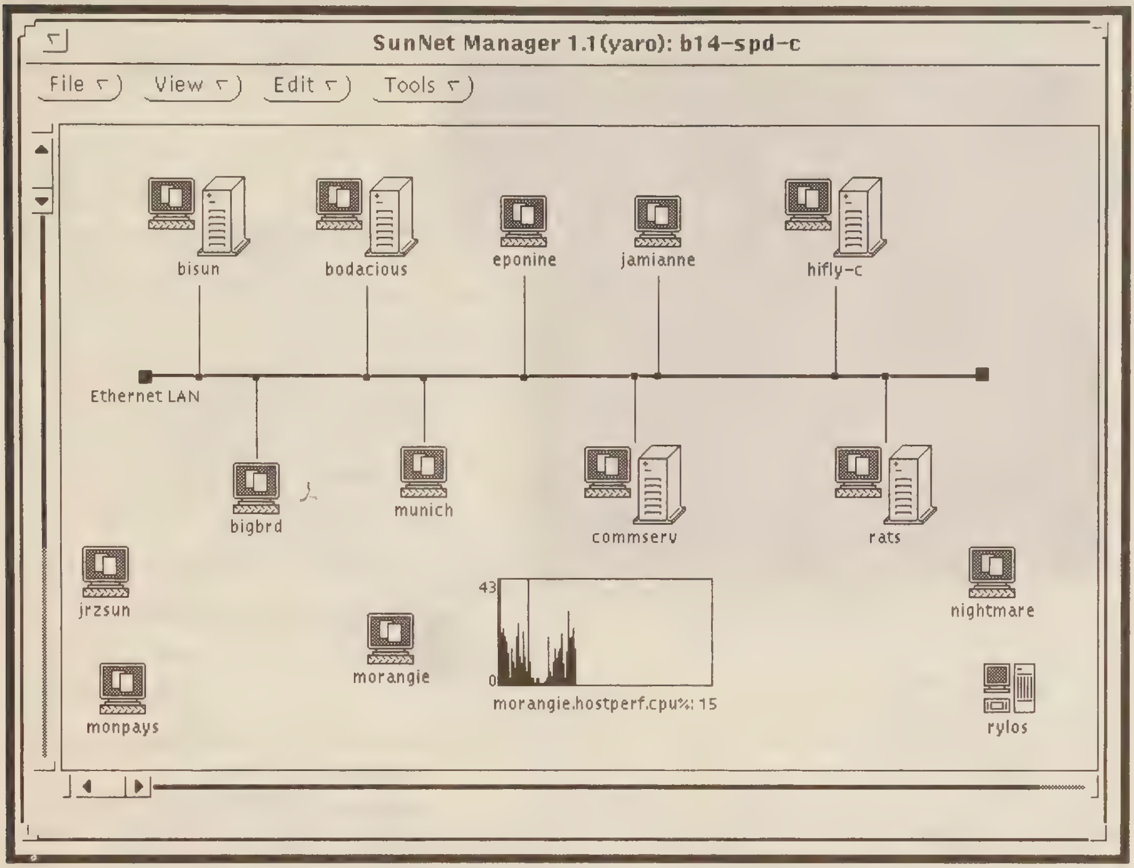
15-14 発見された一連のネットワーク



15-15　Morangieの定義

ストファイルの形の設定オプションを読み込むか，発見ツールのどちらかによって行われる．図15-14は発見ツールを使って発見された一連のネットワークを表している．

ネットワークを発見した後で，ユーザはそれをクリックして，発見されたそれぞれのシステムのグリフを表示する画面を見ることができる．これらのグリフはバス上に一緒に結合することもできるし，浮いているままにしておくこともできる（図15-15参照）．



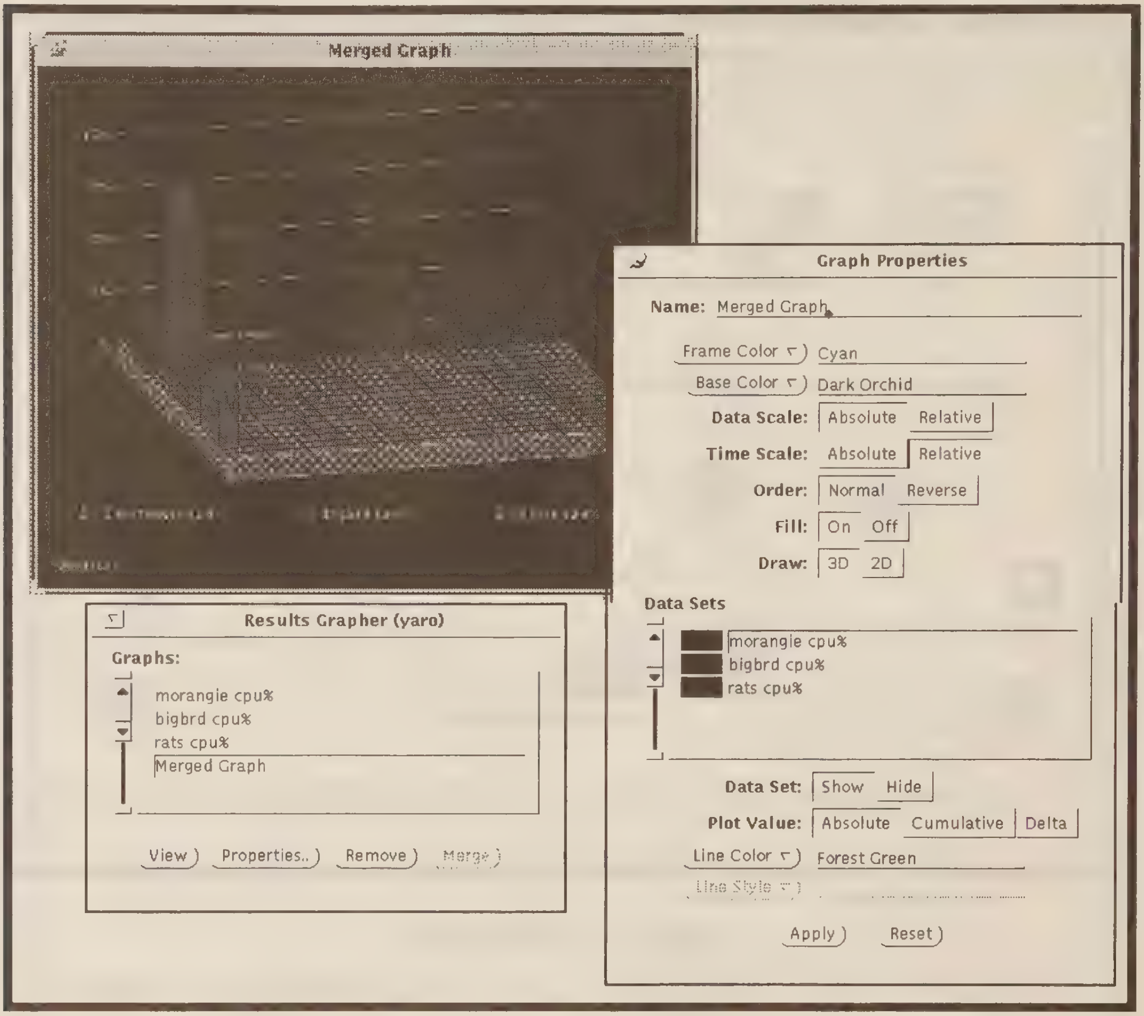
Courtesy of Sun Microsystems

15-16 データ要求の図形的表示

システムが線でつながれているかどうかは，SunNet マネージャの動作においては重要ではない．システムが長時間監視されるのであれば，きれいな図を作ることに意味がある．オブジェクトの図形的な配置替えにより，プリンタ，モデムといったような SNMP をサポートしていないような，新しい，管理されていないオブジェクトを追加することもできる．

それぞれのシステムは，ユーザに提供できるような情報により定義することができる．図 15-15 では，ホスト Morangie の定義が画面上に表示されている．ユーザはアイコンの色を変えたり，このホストに関してどのエージェントを用いるかを指示することができる．

システムに加えて，アクティビティやイベント要求を定義したり，それらを画面上に配置することもできる．図 15-16 はホスト Morangie の CPU 利用についてのアクティビティ要求の例で，画面上で棒グラフで表示されている．イベントは様々な異なる手段で表示することができる．よく用いられるのは，イベントが起こったときに，画面上のアイコンに変化が起きるものである．例えば，CPU 利用率が 60 %に到達すると，アイコンはオレンジになり，90 %を超えるとアイコンは赤になる．CPU 利用率がわからないマシンは黒く表示される．



15-17 いくつかのデータ要求の併合

Courtesy of Sun Microsystems

いくつかのデータ要求を組み合わせて，ネットワークの統一した図形的ビューをつくることもできる．図 15-17 は併合されたグラフの例を示している．3 つの CPU の利用率報告が，それぞれ SunNet マネージャのデータセットとして定義されていたことに注意されたい．

Graph Tool では，複数のデータセットを組み合わせた併合グラフを定義することができる．2 つのホスト (bigbird と rats) のグラフは始まったばかりなので，併合グラフではホスト Morangie のデータが他の 2 つのホストよりも多くなっている．この本では白黒の点で塗りつぶしているが，様々なデザイナーカラーを使うことも可能である．色は，ネットワークマネージャがネットワーク上の重要なイベントを見つける視覚的な手がかりとして利用することができる．

15.9 基本的な管理ツール

SunNet マネージャのほとんどのエージェントは，実際には共通のツールにアクセスするための統一的な手段である．例えば，すてきなコンソールが使えるようになる以前には，多くの管理者が ping コマンドを使用していた．これはネットワーク上のノードへの到達可能性を調べるために，単に ICMP を使っている．

ping エージェントは，ping コマンドを用いるよりしゃれた方法である．コマンドを手動で実行するのでなく，一連のホストに対して定期的な ping を行い，結果をデータベースに記録することを指定したアクティビティ報告を発行することができる．これは，もともとの，コマンド形式の ping コマンドが役に立たないといっているわけではない．ユーザが管理者に電話をかけてきて，サーバがダウンしていると文句をいってきたときには，ほとんどの管理者はマウスをコンソールウィンドウに移動し，"ping servername" と入力して，件のホストがクラッシュしているか（または，存在しないか）どうかを確かめようとするだろう．

```
% ping elvis
elvis is alive
% ping elvis
no answer from elvis
```

第 2 の基本的なコマンドは ifconfig であり，ネットワークインターフェイスを設定したり，情報を得たりするのに用いられる．管理者はインターフェイスが動作しているかどうかを調べられるばかりではなく，どのようなパラメータを使用中であるかを調べることもできる．例えば，イーサネットカードの場合には，ブロードキャストを受け付けるかどうかを知りたくなるかもしれない．もし受け付けないのであれば，ネームサーバの結合づけのようなブロードキャスト要求になぜ答えてくれないのかが，それで説明できるだろう．

より進んだ勉強のために

Rose, Marshall, *The Simple Book*. Englewood Cliffs, N.J.: Prentice-Hall, 1990. A good introduction to SNMP and SMI.

M. Rose, K. McCloghrie, *Structure and Identification of Management Information for TCP/IP-based Internets*, RFC 1155, May, 1990.

Sun Microsystems, *SunNet Manager 1.1 Installation and User's Guide*, Part No: 800-5512-10, Revision A of 1 February 1991.

———, *SunNet Manager 1.1 Tutorial—How to Write an Agent*, Part No: 800-5514-10, Revision A of 1 February 1991.

———, *SunNet Manager 1.1 Tutorial—How to Write an Manager*, Part No: 800-5513-10, Revision A of 1 February 1991.

V. Cerf, *Report of the Second Ad Hoc Network Management Review Group*, RFC 1109, August, 1989

[illegible]

[illegible]

Chapter 16

相互運用性

16.1 概要

かくしてやるべき仕事はやり終えた.ネットワークは完全なものとはならないかもしれないが,ほとんどの問題はユーザよりは手前で解決されるだろう.ユーザがマニュアルをより多く読み始めたなら,あなたが教えることができるより多くのことを質問し始めるだろうが,人間の特性自身がそうであるように,事態は安定した状態になっていくであろう.しかし,安定した状態が続くのも,管理者があなたを呼び出し,1つであったネットワークが2つに置き換えられるまでである.いうまでもないことであるが,2つのネットワークというのは2つの工場島のように独立して稼働することができるが,そうすることはほとんどのMISマネージャの性質に反することになる.

多くの打合せの挙げ句に,最終的にはあなたは2つのまったく異なるネットワークを統合する立場に立たされる.ほとんどの大組織がそうであるように,もちろんネットワークも異なるアーキテクチャをもつ.この章では,これらの異なる計算環境をどのようにして一緒にやっていかせるかを見る.最も簡単な方法は,それらすべてにTCP/IPとNFSでしゃべらせればよいのである.ほんのごくわずかな例外を除いたすべての計算環境は,これらのプロトコルをサポートしている.

もしTCP/IPの経路が選ばれるのなら,それぞれのプラットフォームに適当なベンダを選び,それらをすべて結びつける共通のサブネットワークを選ぶだけの簡単な問題となる.すべての人が規格を同じように読めば,システムは一緒に動くはずである.最後の仮定はちょっとうんざりものであるが.TCP/IPとONC層は両方とも(少なくとも機能の中核部分は)この仮定が妥当というのに充分な時間が経っている.広告されている通りには動作しないという文句(容易に思い浮かぶのはTelnetのネゴシエーションであろう)がときおりあるが,基本のTCP/IPとNFSサービスは異なるプラットフォーム間でも非常にうまく機能する.

単一アーキテクチャが技術的に望ましくなかったり,政治的に不可能だったりすることがある.

幸運にも，このことが実際に問題を引き起こすような場合はほんのわずかしかない．この章では，まず Sun の TCP/IP と NFS ネットワークを IBM の System Network Architecture (SNA) と統合するという，最も大きな挑戦を見ていくことから始める．SNA の相互運用性を見た後に，DECnet から始めて残りの部分に戻って眺める．われわれは Sun の技術を DECnet に統合することや，DEC の技術を TCP/IP に統合することはかなり簡単な問題であることを知るであろう．

次は PC に関する問題である．どんなまっとうなシステムであっても PC にはわれわれが望むオペレーティングシステムとしての基本的なサービスが欠けているので，統合化はより大きな挑戦となる．幸いにも，ほとんどの PC ネットワークのベンダは PC ネットワークがスタンドアローンのものではないということを自覚し始め，TCP/IP 環境への統合の足掛りを渡そうとしている．

この本は TCP と OSI の相互運用性について二，三述べて結ぶ．我々は OSI プロトコルが使用されているのをこの本全体を通して見てきた．この 2 つのプロトコル層は Sun を基本とした環境で利用でき，様々なゲートウェイサービスが利用できる．

16.2 特別なケース：SNA

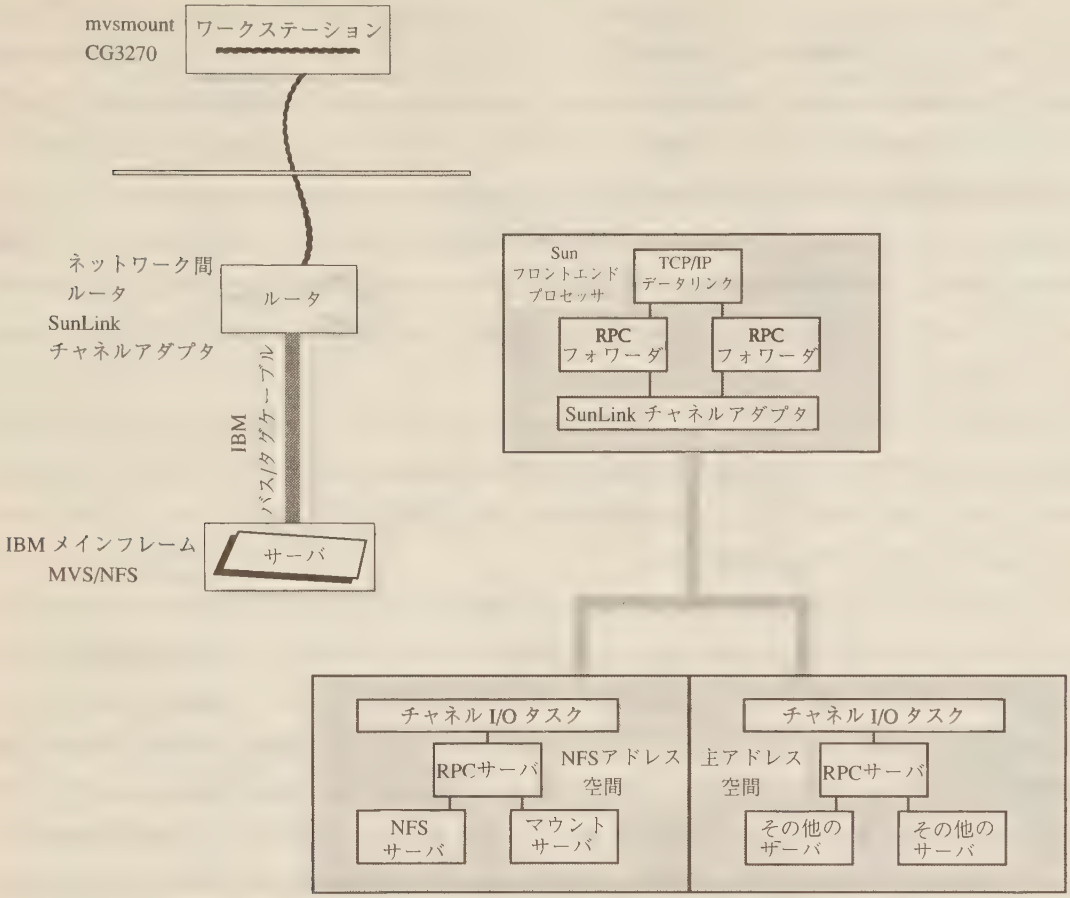
IBM の統合のためのアプローチは 2 つある．簡単な解決法は IBM システムに TCP/IP と NFS をしゃべらせることである．このアプローチは後で見ることにする．いまは，管理上より現実的な SNA をもとにしたネットワークとしてメインフレームシステムを保っていく場合について考えてみよう．

2 つの環境を一緒につなぐにはゲートウェイシステムが必要である．ゲートウェイシステムは様々な異なる構成で動作させることができる．遠隔ジョブ投入 (Remote Job Entry : RJE) のようなソフトウェアを用いて単一のホストに入ろうとする単一のユーザは，どの Sun の上でもローカルなシリアルポートを用いて容易に実行することができる．多くのユーザやデータがたくさんのホストに接続しようとするときには，Bisynch (BSC) や SDLC データリンクプロトコルをサポートしたマルチプロトコル同期ボードである SunNet Multiprotocol Communication Processor (MCP) を使うことができる．メインフレームのチャネルに直接とりつけることによって，より高性能な相互接続ができる．

他のあらゆる機能と同様に，通信は CPU の資源を消費することを忘れてはならない．低速な通信路であるローカルなシリアルポートから流れる一定のデータ通信ですら，Sun-3 の CPU の 5% を消費する．高速な通信路ではより多くの CPU 資源を消費する．1 本の 38.4kbps 回線は Sun-3 の 40%を使う．

16.2.1 SunLink チャネルゲートウェイ

2 つの環境の間で通信する最も高速な方法はチャネルゲートウェイである．IBM チャネルはと



16-1　Sun チャネルゲートウェイ

ても高速なインターフェイスであり，他のホストや通信コントローラや他の高速な周辺機器との接続によく使われる(図 16-1)．チャネルゲートウェイは 2 つの VME カードスロットボードでできていて，バスとタグのインターフェイスケーブルとチャネルスーパーバイザのソフトウェアがついている．そしてこれらにより，IBM メインフレームのブロックマルチプレクサチャネルに接続できる．

1 つの IBM チャネルは 256 個のデバイスまでのアドレス空間をもつ．8 個までの物理的に独立したコントロールユニットか，16 個までの論理的に独立したコントロールユニットがチャネルに接続できる．ゲートウェイとして動作する Sun のサーバは，これらのコントロールユニットの 1 つまたは複数(最大数まで)をシミュレートすることができる．もしマルチプルコントロールユニットをサポートするのならば，それらは同じ種類のコントロールユニットであっても，違う種類であってもよい．

それぞれのコントロールユニットのために，コントロールユニットエミュレーションと呼ばれるアプリケーションが動作する．典型的なコントロールユニットエミュレーションは SunLink Local 3270 であろう．コントロールユニットエミュレーションは，それぞれ定義ファイルとエミュレーションプログラムをもつ．定義ファイルはコントロールユニットの使うチャネルコマンドとその機能に基づくクラスへのグループコマンドをもつ．

チャネルゲートウェイのバージョン 7.0 は交換の機能をもっている．これは大量のファイル転送やジョブ発行，ユーザとの会話的メッセージなどを行うために IBM NJE プロトコルをチャネルゲートウェイの上で動作させることができる，IBM のチャネル対チャネルアダプタのエミュレーションである．

チャネルゲートウェイは他のデバイス(他のチャネルゲートウェイを含む)と同一のチャネルに共存することができる．これは Sun-3 や Sun-4，SPARCstation(4 個以上の 9U スロットと，ローカルディスクと最低 4Mbyte のメモリが必要である)の上で動作する．マルチプルコントロールユニットはより多くのメモリを必要とする．

チャネルゲートウェイは最大 3Mbyte/秒のバンド幅をもつ．NJE のボトルネックは，中に入ってくるファイルのスプールと，外に出ていく送信のキューイングである．最大の高速性能は，速い CPU(Sun-4 など)と速いディスク(SMD ベースのドライブなど)によって達成される．

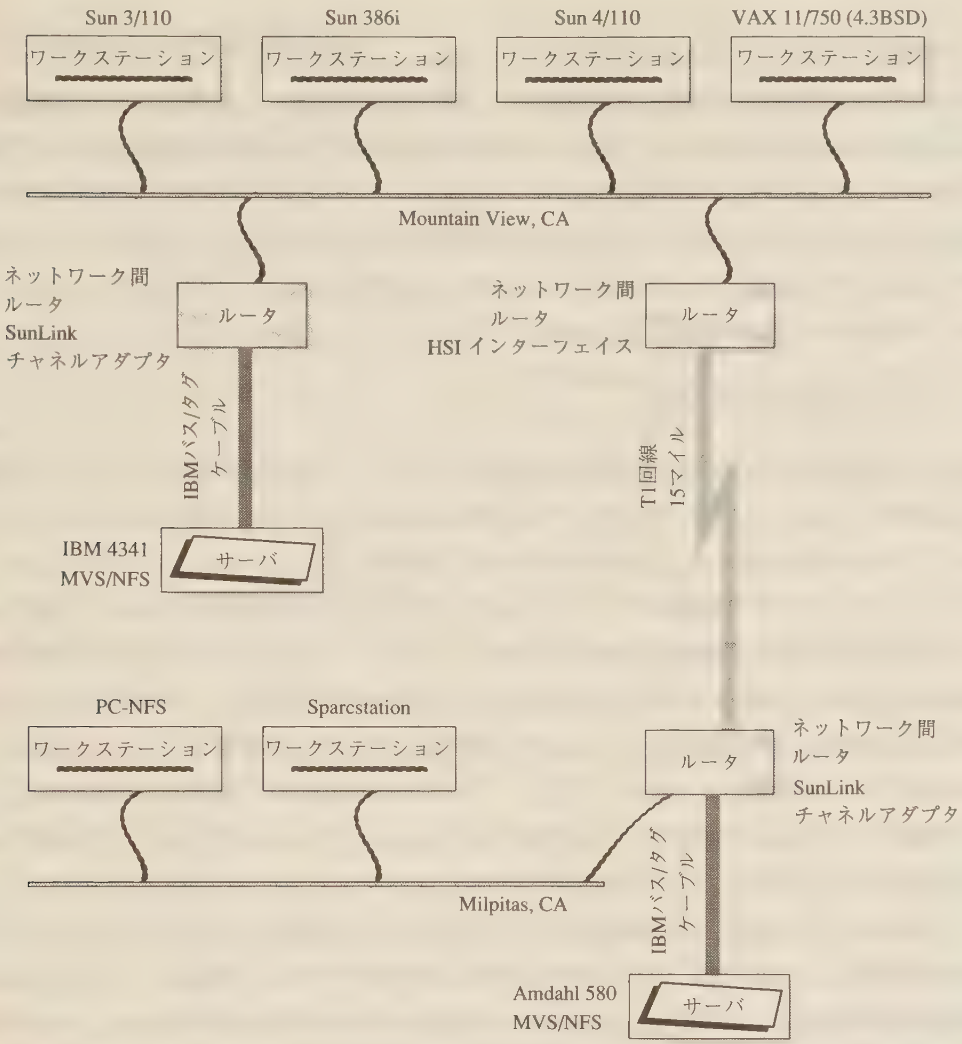
図 16-2 では Sun(そして他のどんな TCP/IP でも)クライアントを SNA の環境に接続する構成の例を示している．この構成は Sun における実在の試験システムの設定である．様々なワークステーションが使われているのに注意してほしい．これらはすべて TCP/IP と NFS が動作している．2 つの大きな IBM システムがある．IBM 4341 と Amdahl 580 であり，両方とも MVS/NFS が動作している．

Amdahl に接続されているチャネルゲートウェイシステムがネットワーク間のルータともなっていることに注意してほしい．15 マイル T1 回線を使ってゲートウェイ/ルータの結合システムと他のルータを接続するのに高速シリアルインターフェイス(High-Speed Serial Interface : HSI)が使われている．図のワークステーションはすべて MVS に基づくサーバのすべてをマウントできる．

16.2.2 SunLink 3270 Products

いくつかの構成の例に戻る前に，まずゲートウェイ上で利用できるソフトウェアサービスをいくつか見てみよう．MVS/NFS は 6 章で述べられている．他の基本的なサービスは 3270 端末のエミュレーションである．3270 端末をエミュレートする 3 つの異なった方法がある．

・SunLink Local 3270
・SunLink SNA 3270
・SunLink BSC 3270



16-2 統合化ネットワークの例

local 構成はチャネルゲートウェイ上で動作する．SNA と BSC の構成は，SNA や BSC を用いてメインフレームに接続する（チャネルゲートウェイでない）ゲートウェイのワークステーションのためのものである．基本的な考えは 3274 のクラスタコントローラをゲートウェイでエミュレートして，どこかにある IBM3287 プリンタと同様にワークステーション上の 3278 端末をサポートするというものである．仮想端末とプリンタは，TCP/IP に基づくネットワーク間のどこにおいても動作させることができる．

加えて，プログラマは 3270 Datastream Application Interface と IBM 3270 PC 互換なファイル転送モードを利用でき，それらは VM と MVS の両方のオペレーティングシステムにおいて動作

する．

1つの変種はTN3270（Telnet 3270のこと）である．これは3270ターミナルエミュレーションをTCP/IP Telnetプロトコルの上で可能にする．TCP/IPがホスト上で動作している必要があり，Ethernet Attachment boxがメインフレームにも取りつけられていなければならない．TN3270はチャネルゲートウェイのよい代替品であり，メインフレームにTCP/IPをしゃべらせつつ同時に3270エミュレーションをサポートしている．TN3270が用いられても，ゲートウェイシステムの役割はまだある．すなわち，バックグラウンドのファイル転送やピアー間プログラムは，メインフレームとSNAプロトコルで対話しなければならない．

16.2.3　SunLink CG 3270

SunLink CG3270ソフトウェアはIBMカラーグラフィックス端末のエミュレーションを可能にする．厳密にいえば，このソフトウェアはIBM 3179Gディスプレイ端末をエミュレートする．このソフトウェアはNeWSウィンドウシステム上で動作する．

このソフトウェアは命令駆動のデバイスであり，仮想端末に曲線やベクトル，テキスト，領域などの絵の要素の記述の集合が送られて動作する．ワークステーションはそれらの要素をPostScript言語に変換し，NeWSウィンドウに送る．PostScriptに基づくウィンドウシステムに内蔵されていることの素晴らしい特徴の1つは，ユーザが大きさを自由に制御できることである．ウィンドウは動的にサイズを変えることができるし，ウィンドウの中身は自動的に調整される．ウィンドウの内部のグラフィックオブジェクトを取り出すことができ，他のウィンドウに動かしたり，局所的に操作したりできる．

いくつかのユーティリティウィンドウによって，ユーザは端末エミュレーションウィンドウの情報を制御することができる．カラーマッパは3270端末の8色をSunワークステーションの1670万色に割りつけることができる．キーボードマッパはIBMの機能やキーストロークをSunのキーボードに割りつけることができる．ファイル転送ユーティリティは3270-PC File Transfer Programと呼ばれるIBMのプログラムを使う．3番目のユーティリティはファイルやプリンタやその他のデバイスにスクリーンダンプを行うための出力制御ウィンドウである．

CG3270ソフトウェアはDatastream Access Interface(DAI)プログラミングインターフェイスの上で構築されている．そのため，SunLinkのゲートウェイ製品の1つ(SunLinkSNA 3270, SunLink BSC 3270, SunLink Local 3270)が利用可能でなければならない．CG3270のライセンスはSunNetライセンスサーバの上で構築されている．ユーザのソフトウェアに対する料金は，ユーザの望む同時に実行されるCG3270セッションの数に基づく．

16.2.4 SunLink BSC RJE

Bisynch(BSC)は非常に古いIBMネットワークにつけられた名前で,かなり昔にSNAにとって代わられている.実際Bisynchはあまりにも古いので,物理層とデータリンク層のわずか2つの層を定義しているだけである.Bisynchの上で構築されるのは遠隔ジョブ投入(Remote Job Entry)機能などのわずかの原始的なサービスである．それではなぜSunがBSC/RJEのためのインプリメントをサポートするのだろうか？　なぜそれらがこの本で述べられているのであろうか？

BSC/RJEは技術の長寿の証明である．これはいまだ最低限の共通のインターフェイスをいくつかのメインフレームシステムに供給している．ここに興味深い3つのサービスを挙げる．2780と3780とHASPである．これらはすべてMVSとVMのどちらのオペレーティングシステムでも動作する．基本的なサービスはカードデック(card deck)を使った遠隔ジョブ投入である．Sunのゲートウェイは,カードリーダと適当な制御やプログラムやデータのカードのついた80文字のレコードのカードデックをエミュレートする．カードリーダのエミュレートに加えて，このソフトウェアはカードパンチやプリンタ，コンソールもエミュレートする.

ゲートウェイはジョブキューイングとホストへのジョブ発行に責任をもつ．ゲートウェイは先入れ先出し方式に基づいて複数のジョブをキューすることができる.ゲートウェイがSunワークステーションなので，ジョブを構成しているファイルはネットワーク上のどの場所にも置くことができる.

ジョブはいくつかの異なる方法で受け取ることができる.1つの選択は,RJEゲートウェイが出すすべてのジョブを1つのディレクトリにまとめることである.別の方法は,ジョブが特定のユーザのディレクトリに送られるようにすることである．より洗練された選択は，コメントカードを出力の中に入れて，SunOSのシェルコマンドが後の処理を行うようにすることである．再び，読者はなぜこのように面倒なことをするのか質問するかもしれない．古くて非常に大きなアプリケーションをもつ環境のことを考えてみよう．例えば，企業注文処理システムである.

しばしば，手打ちの穿孔カードによって行われてきた機能のいくらかは自動化することができる．すべてのユーザがSunワークステーション(またはその他のワークステーションかPC)をもっていれば，データ入力はかなり単純化できる．ウィンドウ，入力書式，データの妥当性チェック，文脈を考慮したヘルプなどを加えることができる．一般的なカードリーダでは得られないすべての機能を加えることもできる.

メインフレームのプログラムは一切変更する必要がない．穿孔カードを読み取ったり打ち出したりする従来とまったく同じやり方で操作できる．違いはカードが自動的にワークステーションから生成され，ゲートウェイシステムを通じて送信されることである.

16.2.5 SunLink SNA ピアー間

SunLink SNA ピアー間(peer-to-peer)ソフトウェアは BSC/RJE や 3270 の操作方法に比べてかなり進んでいる．このソフトウェアは，IBM LU6.2 と，ピアー間通信機能によってより分散化されたネットワーク通信を奨励する Advanced Program-to-Program Communication(APPC)規格とに基づいている．SunLink ソフトウェアは，トークンリングの SNA ネットワークに用いられているノード種別と同じ Physical Unit type 2.1 をプログラマがエミュレートできるようにする．

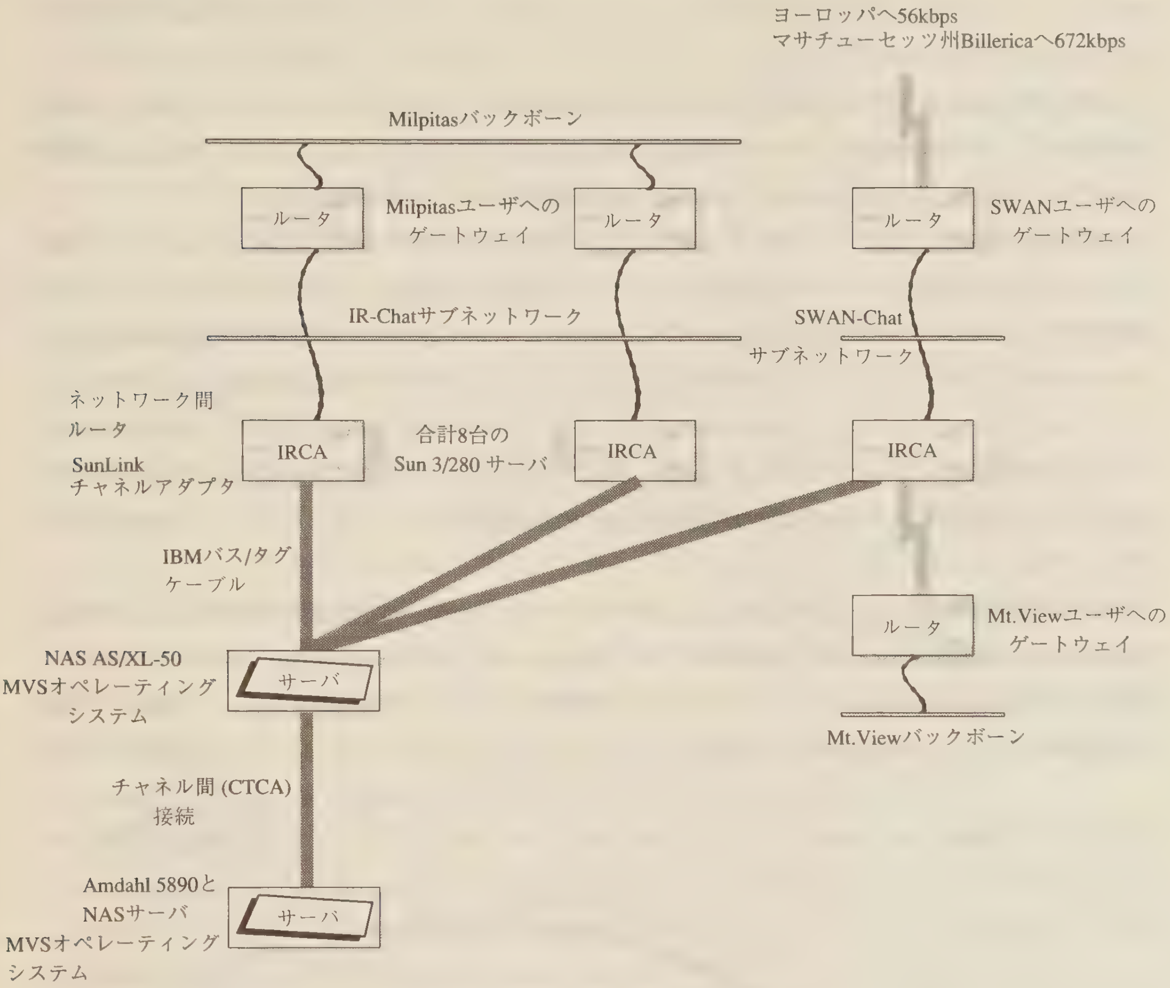
APPC をサポートすることは，メインフレームやその他の IBM の中規模システムのような SNA ネットワーク上のピアーシステム上で動作しているプログラムと，プログラマが対話できるということを意味している．ピアーでは商用のソフトウェアが動作しているかもしれない．また，TCP/IP と SNA の資源の両方を利用できる新たなプログラムを作成することもできる．加えて，このソフトウェアは Document Interchange Architecture(DIA)や，Document Distribution Services(DDS)，Document Library Services(DLS)をサポートしている．DIA/DDS/DLS をサポートすることは，プログラマが IBM メインフレームの上の DISOSS ライブラリシステムを使うことができることを意味している．

典型的な DISOSS の利用は，企業の商用文の記録などの企業規模の情報ファイルである．例えば，保険会社は保険証書を保管するのに DISOSS を使用するかもしれない．DISOSS サービスと対話することによって，Sun のプログラマは Interleaf や Frame の文書処理のようなより進歩したユーザインターフェイスを，企業の情報の蓄積に統合できる．

DISOSS と対話することは SunLink SNA ピアー間 API をなぜ用いた方がよいのかということの 1 つの例にしかすぎない．もっと典型的な例は，分散トランザクション処理システムを構築するために APPC の利点を得ようとする小規模のカスタムなシステムのために見られるであろう．

16.3 SNA ケーススタディ

TCP/IP の世界をどのようにして SNA の世界に接続することができるかを見るために，異なったアプローチをとる 3 つのケーススタディを見る．まず最初は Sun Microsystems で用いられている SunLink チャネルアダプタを用いる場合である．2 番目のアプローチは Town and Country Building Society，英国銀行で選ばれたものである．このアプローチでは，Sun の機器はある環境から他の環境への転送を平滑化するために，現行の MVS に基づくプログラムをエミュレートする．3 番目のアプローチは非常に単純である．IBM に TCP/IP をしゃべらせ，SNA の専門家を 1 ダースかそこら鍛え直すのである．図 16-3 は IBM との接続性(connectivity)を供給するために Sun Microsystems で使われているアーキテクチャを示している．3 つの大きな IBM プラグコンパチブルメインフレーム(PCM)，NAS AS/XL-50 と Amdahl 5890 メインフレームがサーバとして



16-3 SNA接続のためのSWANアーキテクチャ

使われている．これらのコンピュータは製造工程や経理，人事の事務処理に伝統的なアプリケーションである MRP システムを動作させる．データベース管理システムは Cullinet の IDMS である．

接続性はすべてチャネルに接続されたゲートウェイを経由する．物理的な SNA ネットワークやフロントエンドの通信プロセッサや，クラスタコントローラなどはない．システムプログラマがコンソールとして使うためにのみ，少数の 3270 端末があるだけである．

その他のアクセスは IRCA として知られる SunLink チャネルアダプタボードを使い，Sun Internet Router を通して行われる．これらの IRCA システムの 8 個はすべて，Milpitas の設備の 1 つのサブネットワークに取りつけられている．

IRCA システムは 2 つのルータによって共同のネットワークの残りの部分から孤立している．

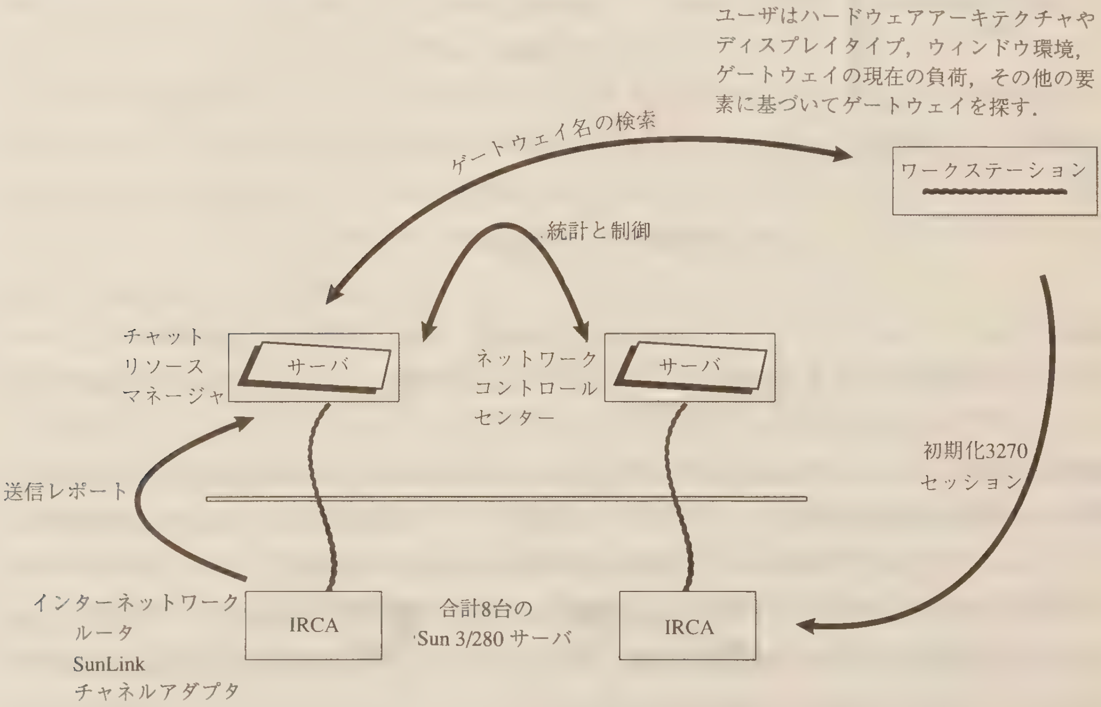
EBay ドメインの残りの部分はこれらの 2 つのルータを通してメインフレームに到達し, IRCA システムの 1 つの上にのる.

外部の Milpitas エリアからのアクセスのために使われる, 小さなサブネットワークの上に孤立しているもう 1 つの IRCA システムがあることに注意してほしい. これらのユーザには, 企業から販売員その他の Sun Wide-Area Network (SWAN) のユーザが含まれる.

これらの外部のトラフィックはすべてこの特別な SWAN サブネットワークを通してくるので, これらは同じ IRCA システムを通る. そして, それらは順番にメインフレームへのチャネル接続を行う. このアクセスの区分けは, セキュリティにおいてよりいっそうの制御を可能にする.

様々に異なるアプリケーションが使用されるが, システムで最も大量に使用されるのはメインフレームのアプリケーションを対話的にアクセスするための CG3270 ソフトウェアを用いたカラー 3270 端末エミュレーションである. ユーザは 3270 端末をもっていないので, 彼らは MVS と対話するのに典型的な 3270 端末より快適な方法である自分のワークステーションの画面を使うことができる.

異なるユーザのアクセスを異なる IRCA ゲートウェイをまたいで分散させるために, 企業の Information Resources スタッフは簡単で小さな SunAccess と呼ばれる RPC に基づくプログラムを書いた. それぞれの IRCA には, いろいろな種類のターミナルエミュレーションに関してある数の同時セッションの能力がある. 加えて, いろいろなワークステーション (または端末でさえ)



16-4 複数のIRCAの負荷均一化

は異なる能力をもっている.

SunAccess は負荷の調整をするために，ワークステーションの能力を利用可能なゲートウェイシステムと照合する．このプログラムの基本操作は図 16-4 に示されている．ユーザがセッションを開始しようとしたとき，ワークステーション上の SunAccess コマンドはチャットリソースマネージャとして知られる負荷均一化サーバに接触する.

資源マネージャは IRCA ゲートウェイシステムからそれらの現在の状態についての周期的な要求を，やはり RPC を使って受けている. 資源マネージャは特定のゲートウェイを次のセッションのために選び，その情報をワークステーションに返す．そしてワークステーションは CG3270 (または他の 3270 ターミナル) エミュレーションパッケージを起動し，ゲートウェイでのセッションを開始する.

興味ある点は，CG3270 ソフトウェアはユーザにとって最も一般的であり (少なくともカラーワークステーションのユーザにとっては)，あるグループの人々のために資源を保守する方法としてライセンスサーバの利用をも可能にすることである．ライセンスサーバは認証とソフトウェアの最大同時ユーザ数の問題を扱う．資源マネージャは最もよい性能を与える IRCA を選び出す.

この負荷均一化システムは非常に効率がよい. 図 16-5 は (最低 9 IRCA が運転中の) システムの負荷均一化レポートの一部を表している．IRCA システムのすべてのバンクがピーク時の約 75% の容量で動作していたことに注目してほしい. 様々な MVS/NFS やプリンタタスクに加えて，このIRCA のバンクは 1 日に約 1000 人の 3270 ユーザを扱うことができる.

特に有益な情報を与えるものは，それぞれの IRCA のレポートの最後にある詳細使用率の部分である．各 IRCA システムのピークがほぼ同じ時間に同じ回数起きていることに注意してほしい. 例えば，IRCA2 は 13 時 26 分に 146 モデル 2 エミュレーションのピークに到達するが，一方，IRCA2 は 13 時 32 分にピークの 147 に到達している.

プリントもこれらのシステムの上に分散される. IBM システムは様々な遠隔プリンタに出力することができる．これらのプリンタのほとんどは Sun ワークステーションに近い場所に置かれ，TCP/IP に基づく Sun サーバに接続されている. IBM システムはゲートウェイを通してすべてのプリントジョブをスプールするようになっている．そして最後には Unix ベースのプリンタサーバに到達する．そこで，プリントデーモンはプリンタサーバに直接つながっているプリンタか補助プリンタサーバにファイルを送る．このシステムを扱うために，たくさんのプリンタがターミナルサーバに接続されている．このターミナルサーバはネットワークを経由してプリントサーバとの間でセッションを維持する.

データベースやファイルサーバへ毎晩スプールされている法人のデータは莫大なものになっている．このデータは分散制御システムや人的資源アプリケーションのようなアプリケーションに使われる. データをワークステーション環境に移動することにより，基本的な 3270 ターミナルの場合と比べて，よりいっそう多くの生産的な道具が開発され，使われることになる.

```
Wed Jan 23 23:57:03 PST 1991
              Percentage of IRCA Utilization for 01/23/91
              Maximum usage = 1014                Limit = 1344
 0    10   20   30   40   50   60   70   80   90   100
HR+    |    |    |    |    ||   |    |    |    |    |
00+##    3
01+##    4
02+#     2
03+#     1
04+##    4
05+###### 9
06+##17########
07+##33###################
08+##54##################################
09+##68###########################################
10+##73###############################################
11+##73###############################################
12+##73###############################################
13+##75################################################
14+##74################################################
15+##70#############################################
16+##66##########################################
17+##55##################################
18+##43##########################
19+##35#####################
20+##30#################
21+##24#############
22+##16#######
23+###### 9

  Usage for : irca2a
    Model 2 usage : (192 available) from 1 @ 02:11  to 146 @ 13:26
    Model 4 usage : (4 available) from 0 @ 00:01  to 2 @ 10:01
    Model 5 usage : (48 available) from 0 @ 00:36  to 4 @ 09:41
  Usage for : irca3a
    Model 2 usage : (192 available) from 1 @ 02:08  to 147 @ 13:32
    Model 4 usage : (4 available) from 0 @ 00:02  to 3 @ 10:22
    Model 5 usage : (48 available) from 0 @ 00:02  to 4 @ 11:02
  Usage for : irca4a
    Model 2 usage : (192 available) from 2 @ 02:08  to 147 @ 13:32
    Model 4 usage : (4 available) from 0 @ 00:02  to 3 @ 08:47
    Model 5 usage : (48 available) from 0 @ 00:02  to 4 @ 10:58
  Usage for : irca5a
    Model 2 usage : (192 available) from 1 @ 02:11  to 146 @ 13:12
    Model 4 usage : (4 available) from 0 @ 00:01  to 3 @ 10:32
    Model 5 usage : (48 available) from 0 @ 00:01  to 4 @ 11:22
  Usage for : irca6a
```

16-5 Sun IRCAのための負荷均一化レポート

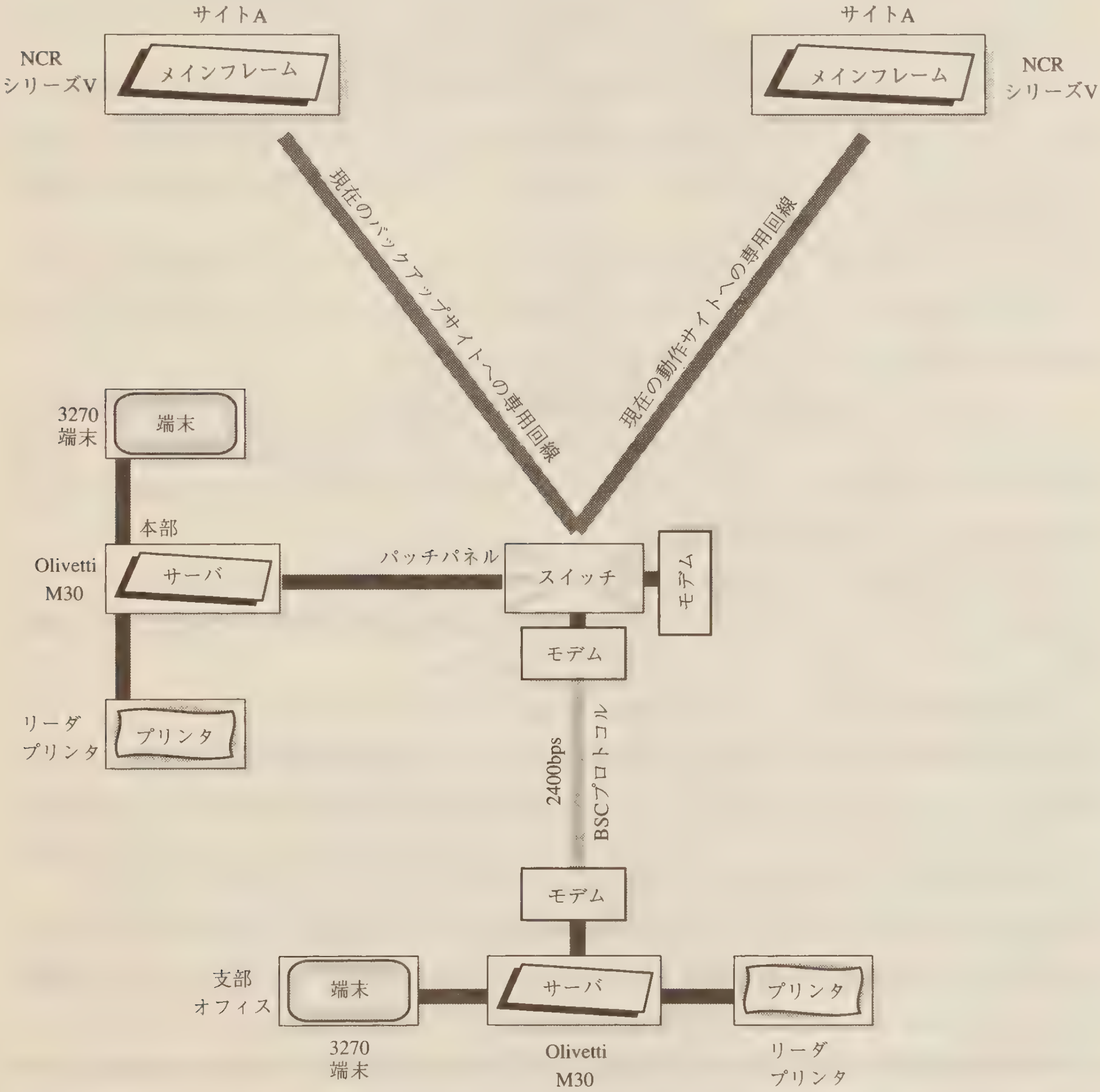
16.3.1 Town and Country

IBM の接続性への別のアプローチは英国にある Town and Country Building Society で用いられている．Building Society は，支払い能力を何とか残しているという明らかな違いを除けば，米

国の貯蓄貸付産業と似たものである.

80 年代の半ば，この組織は NCR series V メインフレームシステムの 3270 端末に基づく非常に伝統的なアプリケーションを使っていた. 2 つの主要処理サイトは，一方のサイトが活動しているときはもう一方がバックアップとして働いて，莫大な量の処理を提供する．両方の施設に操作上の経験を与えるために，それらは周期的にメインサイトの役割を切り替える.

図 16-6 はこの環境のための初期構成を示している．分岐システムのそれぞれの制御は Olivetti M30 プロセッサによって与えられ，それは端末やリーダ/プリンタデバイスに接続される. すべての支部のオフィスは本部へ 2400 bps のモデムと BSC/RJE プロトコルを使用していた.



16-6 Town and Country, フェーズ1

本部ではすべてのトラフィックを結び付け，適当な処理サイトに送るようなパッチパネルがある．1台のメインフレームが故障してもトラフィックはバックアップサイトに再び経路選択される．

この構成の問題は，M30プロセッサが現在の最新のプロセッサではないことである．目標は，支部をより標準的な装備に移行することと，(ワードプロセッサなどの)より最新のアプリケーションを配給することである．

このような環境で同じくらいに重要な目標は現在のアプリケーションを走らせ続けていくことである．このことは銀行に関してはかなり重要である．というのは「新しいコンピュータソフトウェアにバグがあるから」といって顧客が自分の金を引き下ろせない場合を考えてみればよい．このような状況はほとんどの金融業界において最悪のものと考えられている．

図16-7は，より新しい環境を作り上げるためにどのようにして段階的なステップが踏まれたかを示している．Sunサーバの上で動作する3270端末をエミュレートするRPCに基づくプログラムが書かれた．そしてPC-NFSが動作する一連のPCシステムが，出納係や貸付事務長や支店のその他の従業員のために配給された．

PCシステムは基本モードでは単純な3270エミュレーションソフトウェアが動作する．このソフトウェアはSunサーバへのトランザクションを発行する．そしてサーバはトランザクションを前と同じBSC/RJEプロトコルを用いてメインフレームに送る．

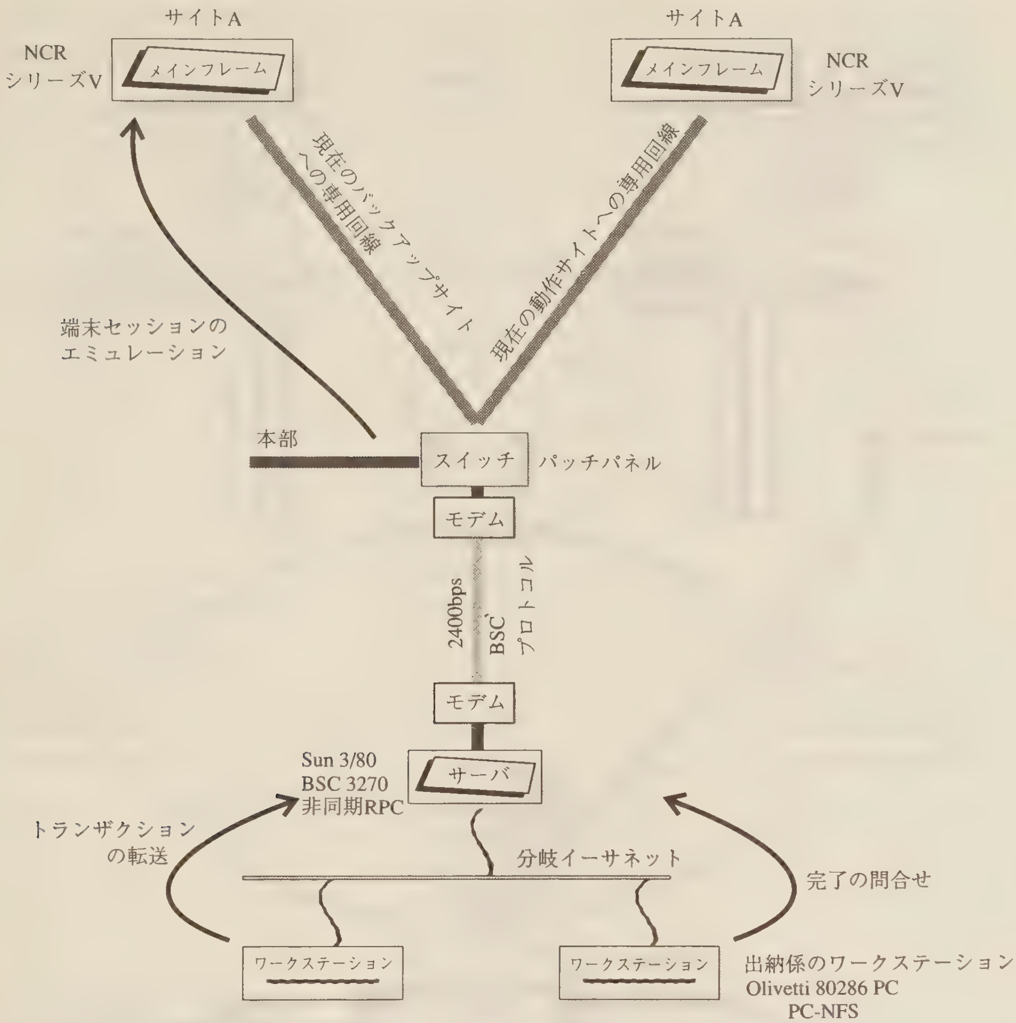
サーバはそれぞれの端末に対してステータス領域を確保した．ワークステーションが未解決のトランザクションをもっているときは周期的に完了を問い合わせ，結果をとり出す．サーバには状態がないことに注意せよ．つまり，未解決のトランザクションがあるかどうかを記憶するのをPCの責任とし，エラー回復を単純化した．

このアプローチはモジュラであることにも注意してほしい．故障したときのためマルチドロップ回線のためのパッチパネルからは，システムは既存の環境に透明である．支店は1つずつ置き換えることができ，既存のシステムと共存することができる．

この段階の直接の利点は，ユーザの端末がLotusやワードプロセッサなどのアプリケーションを実行できるPCであることである．各出納係のシステムは20Mbyteの内部ディスクと1.44Mbyteの3.5インチフロッピーディスクをもつOlivetti 290 PCやIntel 80286 PCである．それぞれのPCはVGAカラーモニタ，1Mbyteの基本RAM，4Mbyteの拡張メモリ，シリアルとパラレルのポートをもち，(1988年のPCの基準で)置き場所の小さなシステムと考えられていた．

面白いことは，これらのPCシステムが問題への他のアプローチとして以前に考えられていたことである．それは成功しなかったが，そのシステムではdbaseに基づくアプリケーションが独自のゲートウェイとPCネットワーク方式を用いていた．

次の段階では図16-8に示すようにBisynchプロトコルから離れ，データ通信のより新しいアプローチに移行する．端末をエミュレートするプログラムは主処理サイトまで移動され，Sun-3/470

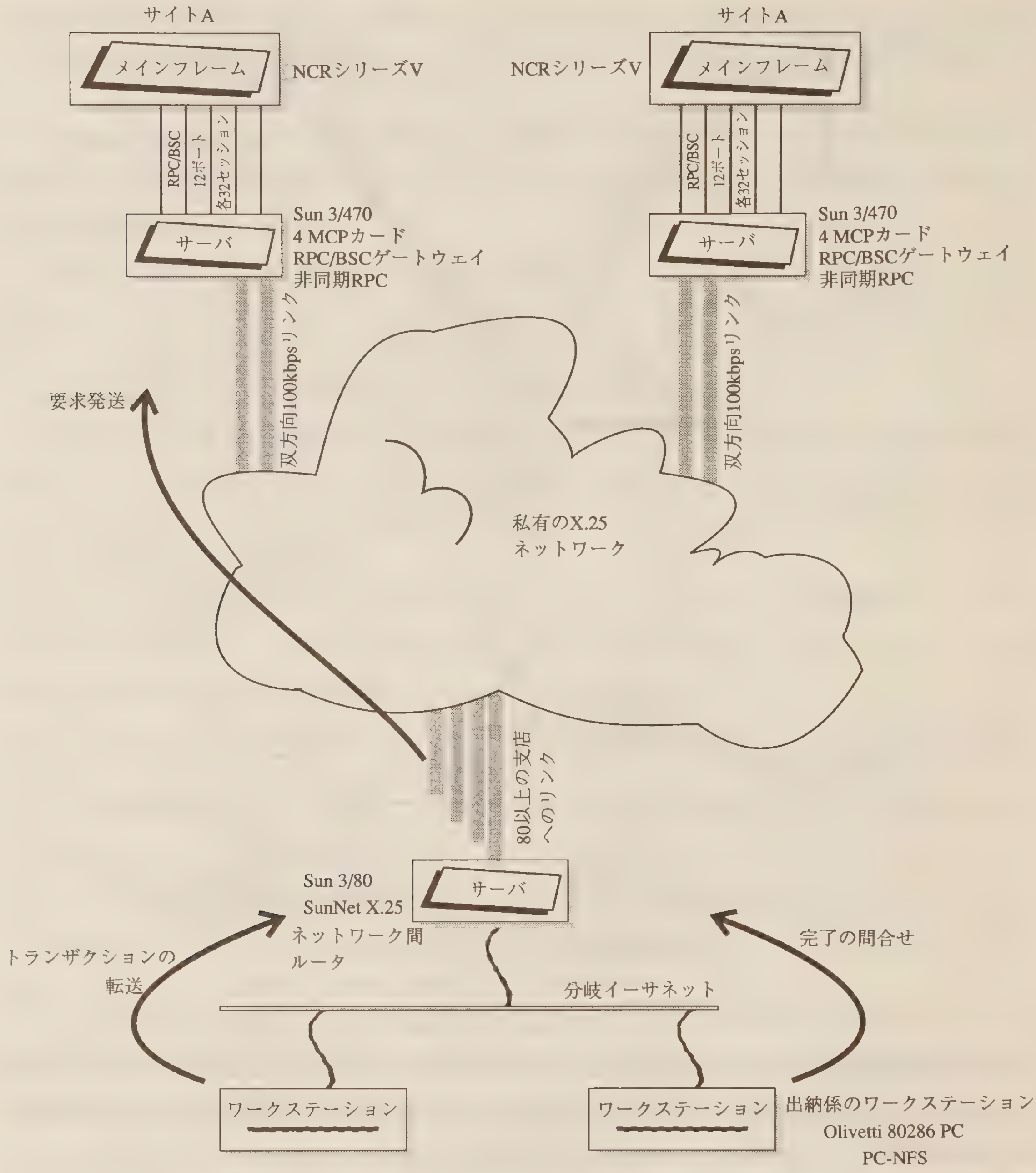


16-7 Town and Country, フェーズ2

の上に移動された．このサーバシステムは，X.25 ネットワークへの 2 つの 100kbps のリンクに加えて NCR メインフレームへの 12 ポートをもっており，それぞれが 32 個のセッションを同時に扱う能力がある．

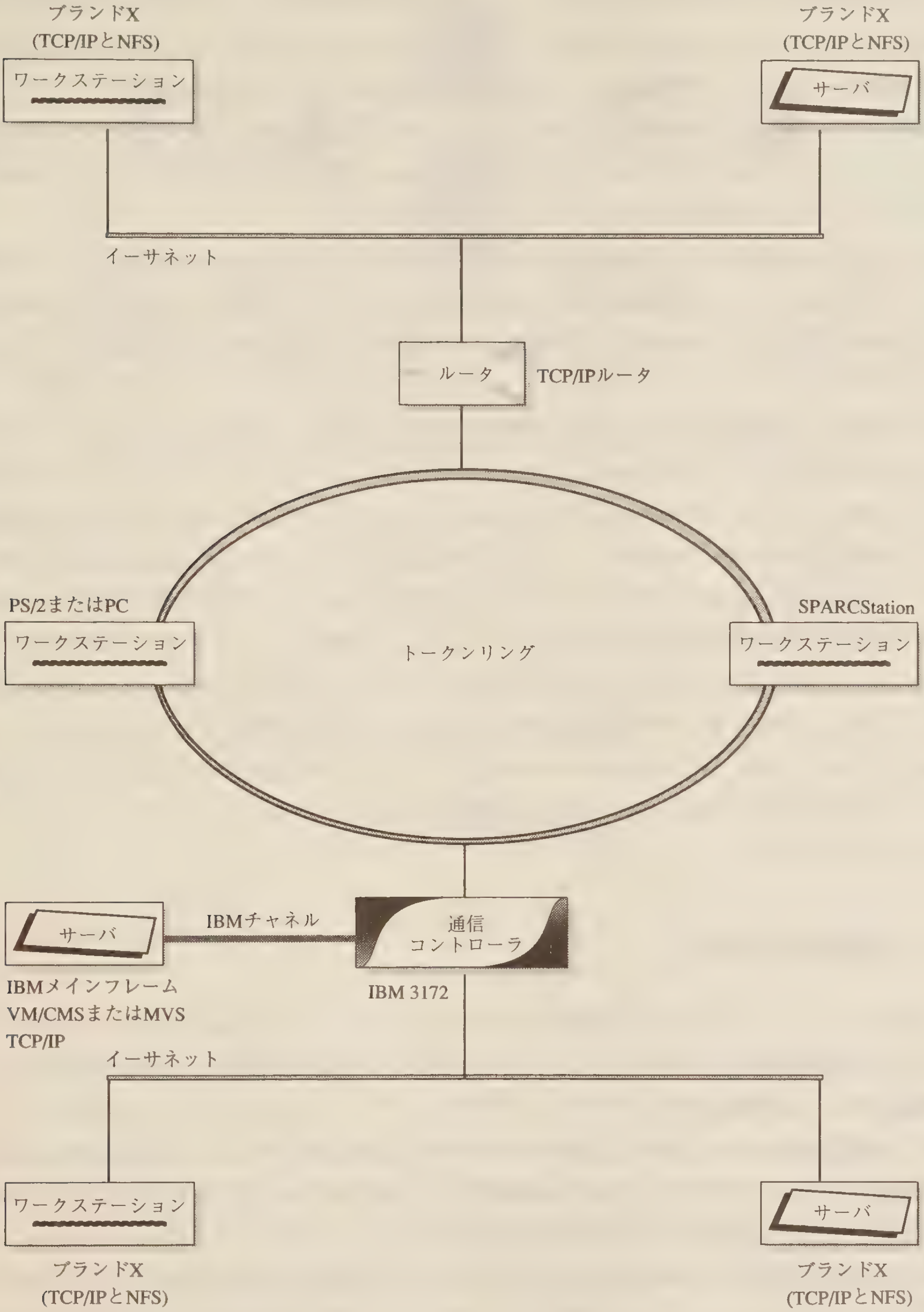
支店のオフィスの機器は同じままであったのだが，もはや RPC プログラムを実行しなくなった．その代わりに，この支部のサーバは SunNet X.25 ソフトウェアを伴った単純なネットワーク間ルータとなった．ワークステーションのソフトウェアは変更する必要がなかった．単に異なるサーバのアドレスを使うようになっただけである．

この時点ではメインフレームシステムの動作に変わりはないが，支店への分散はかなり高速になり，より自由度が高くなる．次の段階は図には示さないが，NCR Series V のサービスをより新



16-8 Town and Country，フェーズ3

しいアプローチで追加することである．これには Sybase リレーショナルデータベースソフトウェアを実行する Sun-3/490 を加えることが含まれる．Sun-3/470 はデータ処理ではゲートウェイとして働き続け，要求を Sybase サーバまたは NCR メインフレームに転送する．Sybase システムは貸付処理などのアプリケーションに使うことができる．



16-9 TCP/IPを用いたIBMの統合

ここでの鍵となるのは既存のネットワークをそのまま継続できたことであり，移り変わりが緩やかに行われたことである．支店ではゆっくりと切り替えることができた．それが完了した後に，データ処理ゲートウェイが移動され，最後にUnixに基づくサーバが加えられた．

このシステムが無から構築されたのであれば(大きな金融機関ではそうなることは滅多にない)，すべてのシステムはTCP/IPプロトコルと，VMやMVSオペレーティングシステムのいずれかにおけるNFSの実装に基づくものになったであろう．

図16-9はこの最後に述べた接続性の可能性を示している．メインフレームシステムはIBM 3172クラスタコントローラに接続され，それが今度はトークンリングやイーサネットに接続されることに注意せよ．この点からすべての環境は均質であり，IBMは単にもう1つのサーバにすぎない．

SunのIBMとの接続性はMVS/NFSの開発において一体となる部分を演じてきた．このプロジェクトの技術者はすべて机にSunワークステーションをもっていた．メインフレームとの対話には3270端末エミュレータかNetwork Job Entry(NJE)ソフトウェアが用いられた．3270端末はシステム作業をするためにMVSのコンソールを獲得するのにしばしば使われた．NJEに基づく遠隔ジョブ投入はソースコードをメインフレームにアップロードするのに使われたり，Unixのmakefileの制御のもとでMVS/NFSのアプリケーションを構築しようとするときに使われた．

ソースコード制御までメインフレームから遠ざけ，UnixのSCCSユーティリティを代わりに用いた．これには370アセンブリ言語の6000行も含まれている．開発チームは，MVSのルーチンをUnix上でエミュレートし，メインフレーム上でデバッグされる必要のあるアセンブリ言語のルーチンだけを除いて，高レベルのMVSの機能がワークステーション上でデバッグできるようにするまでに至った．

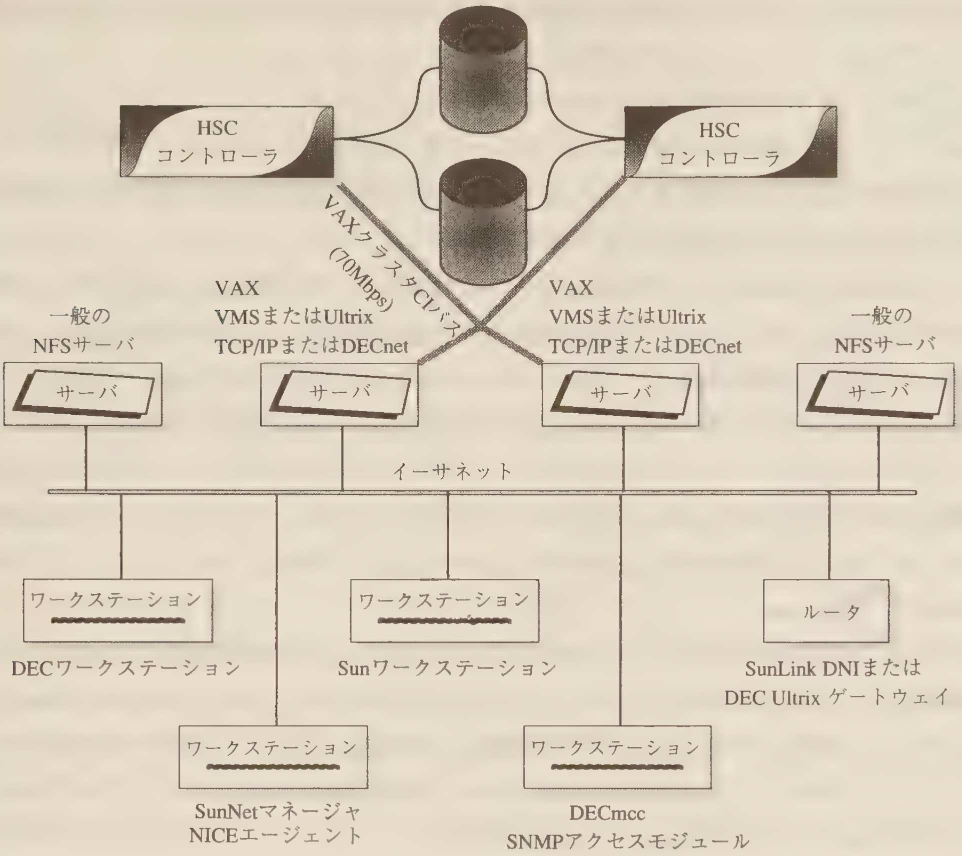
16.4　世界の残りの部分

いまやメインフレームシステムも邪魔にはならないので，ここでは他のいくつかのものをネットワークの中に統合することを見ていこう．例えばDECとSunの機器は，しばしば同じネットワーク環境の中に一緒に存在する．

DECの機器はDECnetに基づく固有のネットワーク環境をもっている．DECnet Phase IVは独自の環境であるが，TCP/IPとは多くの類似点をもっている．例えば，DECnetはファイル転送プロトコルと重畳するData Access Protocol(DAP)をもっている．

2つの環境の統合はいくつかの階層で行われてきた．DECnetとSunの機器はFDDIやイーサネット，Ultranet networks，X.25を含む同一のデータリンクを共有することができる．並行するネットワークを動作させることが可能であり，下位の媒体を共有するだけである．

図16-10は4種類の機器をもった小さなネットワークの構成である．



16-10 DECnetとの相互接続

- ワークステーション
- ネットワーク管理コンソール
- サーバ
- ゲートウェイシステム

並行した世界を動作させる場合には，ゲートウェイシステムは用いられない．DEC ワークステーションは DECnet を使い，VAX システムと会話する．DECnet のノードは互いに通信できるだけであり，DAP のような DECnet のアプリケーションを実行する．

他の世界，つまり TCP/IP の世界は Sun ワークステーションおよびネットワーク上にある NFS や TCP/IP のサーバすべてである．ここでも，この世界のメンバーはお互いだけで通信することができ，NFS や FTP のように TCP/IP のアプリケーションだけを走らせるだろう．2 つの世界はイーサネットのもとでサービスを共有していることに注意してほしい．イーサネットは単に目的地にパケットを配送するという一般的な目的のサービス提供者である．

2 つの世界を結ぶためには 2 つのアプローチがある．簡単な方法は DECnet の世界をすべて

TCP/IP に替えることである．DEC は VMS および Ultrix オペレーティングシステムの両方に対して TCP/IP をネットワーキングの選択としてサポートしている．

DEC は VAX クラスタ環境の上での NFS プロトコルもサポートしている．VAX クラスタシステムは，いくつかの異なる大型ミニコンピュータを 70Mbps のバスを介して専用ディスクコントローラに接続することができる．ディスクコントローラは今度は巨大な量の冗長記憶装置を提供するデュアルポートディスクアレイに接続される．

クラスタは非常に大量のデータのための，よい NFS サーバを形成する．6 章で見たこの問題に対する別のアプローチは Auspex や Epoch などの会社が販売している専用 NFS サーバ(または高速なディスクをもつ専用 Sun サーバ)を用いることである．統合化のための 2 番目のアプローチは，DEC システムに TCP/IP と DECnet の 2 つのネットワーク言語をしゃべらせることである．このアプローチを使うと，サーバはそのファイルやプログラムやプリンタその他の資源を両方の環境に供給することができる．最後に，第 3 のアプローチとしてゲートウェイを使う方法を挙げる．ゲートウェイは，普通は単に 2 つのアプリケーションに橋渡しすることによって，2 つの環境をある点でつなぐ．

Sun と DEC は両方とも(他のベンダもやっているように)2 つの環境を相互接続するソフトウェアを販売している．Sun では，ソフトウェアは SunLink DNI といい，DECnet Phase IV の終端ノードのように働き，TCP/IP ユーザが DECnet NICE(ネットワーク管理)や DAP プロトコルのようなプロトコルにアクセスできるようにする．等価な DEC のソフトウェアは Ultrix ゲートウェイで，DECnet のメールやファイル転送，端末エミュレーションソフトウェアを TCP/IP の相当物へとつなぐ．多くの人にとってより重要なアプリケーションの 1 つはネットワーク管理である．DECmcc と SunNet Manager は両方とも他方の環境の管理もサポートしている．

16.4.1 DECnet Phase V

DECnet の Phase V は，独自の DECnet Phase IV プロトコルに OSI および TCP/IP プロトコル族からのプロトコルを追加した拡張である．Phase V は実際には 3 つの異なるネットワークをつなぎ合わせたものである．

ネットワークをつなぎ合わせるやり方は，複数のエントリポイントを与える．例えば，ファイルアクセスのために DECnet ノードは FTAM か FTP か DAP(または NFS かデータベースパッケージ)を実行することができる．

この環境に接続するときの迷路を切り開く簡単なやり方は，共通の地盤として OSI を使うことである．Phase V は FTAM や X.400 のような基本の OSI プロトコルをサポートしている．SunLink OSI と SunLink Message Handling System(MHS)ソフトウェアは，これらの 2 つのプロトコルをサポートしている．そこで共通の地盤として使うことができる．

環境を一緒に接続する他のアプローチは，TCP/IP と NFS を共通の地盤として使うことであ

る. これはユーザが LAT や名前サービス, DAP などの DEC の独自プロトコルに縛られていない限りうまくゆく.

16.4.2 OSI

Sun の機器のための OSI 接続性は, 基本的に OSI プロトコルの主要なサブセットをつまむことにより提供されている(図 16-11). SunLink MHS ソフトウェアは本来の SMTP プロトコルを X.400 の世界につなげるための CCITT X.400 メッセージ処理システムゲートウェイである.

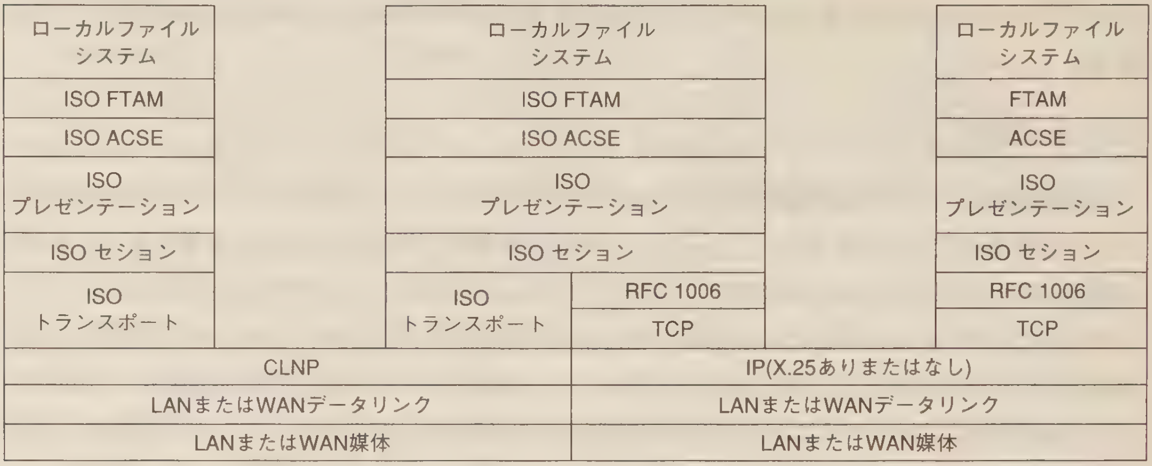
FTAM とその他のサービスは, 基本の SunLink OSI ソフトウェアシステムの一部として提供される. このソフトウェアは X.25, イーサネット, トークンリング, または FDDI データリンクの上で動作する. 他のベンダの OSI の実装に接続するために, データリンクの中で一般的な共通項を探す.

OSI 接続性のその他のアプローチは ISO Development Environment(ISODE) から来る暫定的な解決を用いることである. 図 16-12 は TCP/IP に基づく Internet が OSI のアプリケーションのためのデータを転送するのにどのように使われるかを示している.

4 章を見たように, RFC 1006 では, コネクション指向のネットワークサービスに基づく環境に期待するようなサービスを TCP がどのようにして与えるかが定義されている. もしヘッダ情報に少しの情報を付加するなら, TCP/IP 環境は経路中の単なる 1 ホップと見なすことができる.

<table>
<tr><td rowspan="3">CCITT
X.400</td><td rowspan="3">カスタム
プログラム</td><td colspan="3">ISO 8571
ファイル転送, アクセス, および管理(FTAM)</td></tr>
<tr><td colspan="3">ISO 8650
アソシエーション制御サービス要素(ACSE)</td></tr>
<tr><td colspan="3">ISO 8823
プレゼンテーションプロトコル</td></tr>
<tr><td colspan="5">ISO 8327
セションプロトコル</td></tr>
<tr><td>ISO 8073
トランスポートクラス
0および2</td><td colspan="4">ISO 8073
トランスポートクラス4</td></tr>
<tr><td>ISO 8878
コネクション指向
ネットワークサービス
(CONS)</td><td colspan="4" rowspan="2">ISO 8473
コネクションレスネットワークプロトコル(CLNP)</td></tr>
<tr><td>ISO 8208
X.25 パケットレベルプロトコル(PLP)</td></tr>
<tr><td>ISO 7776
リンクアクセス
プロトコルB(LAPB)</td><td colspan="4">ISO 8202/2
論理リンク制御(LLC)
クラス1および2</td></tr>
<tr><td>X.21 bis</td><td colspan="2">ISO 8802/3
10Mbps CSMA/CD</td><td>ISO 9314
FDDI</td><td>ISO 8802/5
4Mbps トークンリング</td></tr>
</table>

16-11 SunのOSIのサポート



16-12　TCP/OSIの橋渡し

OSI のアプリケーションを TCP/IP の環境で動かすためには ISODE(やこのアプローチの商業的な派生物の 1 つ)などのソフトウェアが使われる．X.500 サービスのようなアプリケーションが OSI/TCP 混成層の上で動作する．

16.4.3　PC

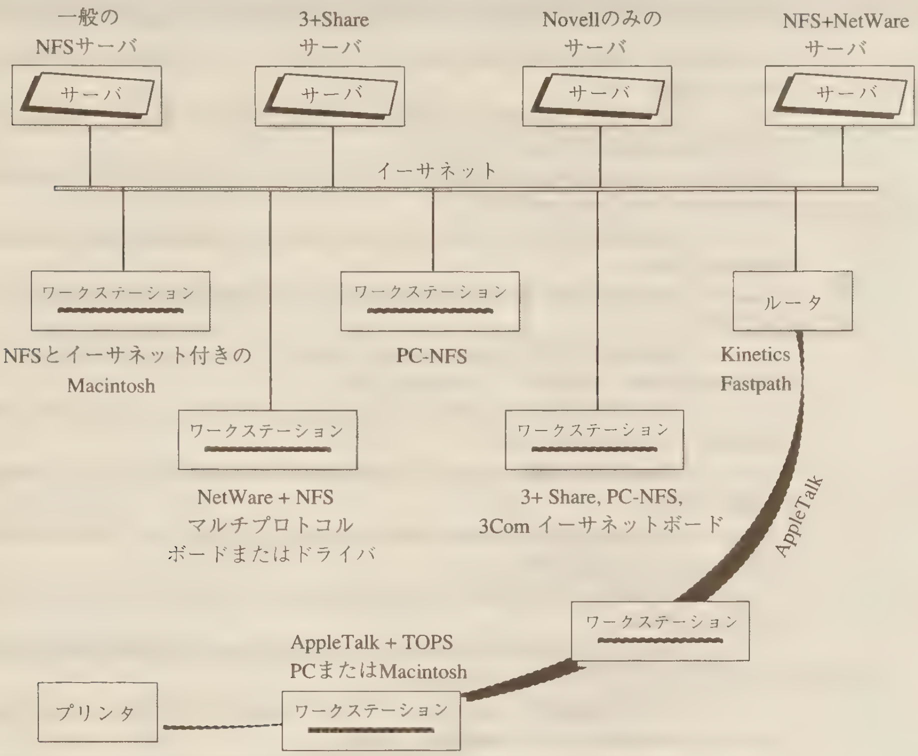
われわれには 1 つ置いてきたものがある．PC である．ここでも簡単な解決は DOS や Macintosh オペレーティングシステムで利用することのできる PC-NFS やその他の TCP/IP に基づく解決法のどれかのような単一のプロトコルを走らせることである．

すべての IBM メインフレームに SNA をしゃべらせるのをやめさせられないのと同じ理由から，不運にもすべての PC システムに NetWare のようなプロトコルを使うのをやめさせることはできない．Macintosh ワークステーションにも同じような選択がある．AppleTalk か TCP/IP である．

以前と同様，並行世界を動作させることができる．例えば，Novell NetWare サーバが Novell データを Novell ワークステーションに供給するのに用いる一方で，並行した TCP/IP 世界がそのまわりをめぐることができる(図 16-13)．マルチプロトコルドライバか(できれば)マルチプロトコルボードを使うことで，PC が両方の世界で同時に動作することは可能である．

NetWare 環境を統合するもう 1 つのアプローチはサーバに両方のプロトコルをサポートさせることである．サーバは Novell Operating System の一部として NetWare Loadable Module を実行する．これにより Novell サーバを一般的な NFS サーバに変える．

Macintosh では，TCP/IP と AppleTalk の両方を動作させるより有無をいわせない理由がある．AppleTalk はすべての Macintosh システムや多くのレーザプリンタを含む多くのデバイスに組



16-13 PCシステムによる相互運用性

み込まれていて，イーサネットよりも一段と安い．

TCP/IP の世界と AppleTalk の環境の間のゲートウェイシステムが存在する．例えば，Kinetics は 2 つの環境を結ぶ Fastpath ゲートウェイシステムを使っている．Tops は AppleTalk と TCP/IP の環境の両方をサポートできるソフトウェアをもっている．

ここでも，より簡単な解決法はワークステーションをイーサネット上に置き，TCP/IP を動作させることである．X-ウィンドウや NFS のようなサービスを利用することで非常に高い透過性を得ることができる．「申し訳ありませんが，その機器上のそのプログラムでそのデータを利用することはできません．」などというような並行世界アプローチによって作られる人工的な壁を心配することなしに，どのようなサーバのどのようなプログラムもワークステーションで動作させることができる．

より進んだ勉強のために

Malamud, *Analyzing DECnet/OSI Phase V,* Van Nostrand Reinhold (New York: 1991).

———, *STACKS: Interoperability in Today's Computer Networks,* Prentice Hall (Englewood Cliffs, NJ: 1991).

Sun Microsystems, *SunLink BSC RJE User's Guide*, Part No: 800-3033-10, Revision A of 18 October 1988.

———, *SunLink CG3270 User's Guide*, Part No: 800-4962-10, Revision A of 26 October 1990.

———, *SunLink NJE User's Guide*, Part No: 800-3009-05, Revision 50, 19 August 1988.

———, *SunLink DNI Programmer's Guide*, Part No: 800-5375-10, Version A, 15 March 1991. DECnet への SunLink 接続.

———, *SunLink DNI System Administrator's Guide*, Part No: 800-5374-10, Version A, 15 March 1991.

———, *SunLink DNI User's Guide*, Part No: 800-5373-10, Version A, 15 March 1991.

———, *SunLink OSI System Administration Guide*, Part No: 800-3214-10, Revision A of 21 July 1989.

———, *SunLink OSI Programmer's Manual*, Part No: 800-3215-10, Revision A of 21 July 1989.

———, *SunLink SNA Peer-to-Peer API Programmer's Guide*, Part No: 800-3045-10, Version A of 7 October 1988.

———, *SunLink SNA Peer-to-Peer DIA Programmer's Guide*, Part No: 800-3044-10, Version A of 7 October 1988.

———, *SunLink SNA Peer-to-Peer System Administration Guide*, Part No: 800-3043-10, Version A of 7 October 1988.

用語集

1003.4
POSIX実時間拡張.

1003.6
POSIXセキュリティ拡張.

1003.7
POSIXシステム管理拡張.

1003.8
POSIXネットワーク拡張.

10BASE2
10Mbps/ベースバンド/200m. 細径ケーブルイーサネットのIEEE標準.

10BASE5
10Mbps/ベースバンド/500m. 太径同軸ケーブルイーサネットのIEEE標準.

10BASET
10Mbps/ベースバンド/撚り対線. 撚り対線イーサネットのIEEE標準.

2780/3780
IBMのBisynchで使用される遠隔バッチ端末のモデル. Bisynchゲートウェイは2780/3780エミュレータとして参照されることが多い.

2進木 (binary tree)
しばしば Btree と呼ばれる. データを頻繁に更新する環境で使われる動的検索指標をもつ記憶構造(storage structure)である.

2進絶対時間 (binary absolute time)
協定世界標準時 (coordinated universal time; UTC), 時差要素 (time differential factor; TDF) および不正確さの項で表現された時間.

3+
3Comネットワーク製品

3090
IBMのメインフレームの最上位機種. シエラとも呼ばれる.

3270
IBM環境で使用される端末のシリーズ.

3270 Display Stations
IBMメインフレームコンピュータの端末.

3274
IBM機器のためのクラスタ制御装置. しばしば端末集線装置と呼ばれる.

3480
IBMのテープカートリッジ

3705
IBM通信制御装置. SNAにおける物理ユニット(PU)タイプ4で, トークンリングネットワーク, クラスタ制御装置, IBM以外のSNAゲートウェイ, その他の装置を, PUタイプ5のホストに接続する.

370アーキテクチャ(370 architecture)
3090プロセッサを始めとするIBMメインフレームコンピュータのアーキテクチャ.

3725
3705参照.

3Com
イーサネット制御装置とパソコンベースのネットワーク設備で知られる通信会社. Bridge Communicationsに合併された.

4010/4014
Tektronixのグラフィック端末のシリーズ. グラフィック端末の事実上の標準.

4.3BSD
4.3 Berkeley Software Distribution. バークレー版Unixの最新版

4GL
第4世代言語を参照.

5250
システム/3Xシリーズ上で使用されるIBM端末.

802.2
論理リンク制御(logical link control)のIEEE標準.

802.3
CSMA/CD(イーサネット)媒体アクセス法(medium access method)のIEEE標準.

802.4
トークンバス媒体アクセス法のIEEE標準.

80x86
パソコンで用いられるIntel製のマイクロプロセッサ群. PC/ATは80286チップを使用している. PS/2のような最近のシステムでは, 32ビットの80386や80486チップが用いられている.

9370
IBMの中型プロセッサ.

ABM
非同期平衡モードを参照.

AC
アクセス制御. トークンやデータパケットに対する優先ビットや予約ビットを格納するトークンリングのフィールド.

ACK
確認(acknowledge). データを受け取ったことを確認するネットワークパケット.

ACL
アクセス制御リスト(Access Control List). オペレーティングシステム中のセキュリティの機能で, 特定のユーザのリストに対して許可される動作のリストとしてオブジェクトのセキュリティを記述することができる.

ACM
米国計算機学会(Association for Computing Machinery).

ACSE
アソシエーション制御サービス要素を参照.

ACSNET
オーストラリアコンピュータサイエンスネットワーク. ACSNETはオーストラリアのコンピュータ科学と学術研究機関のための主要な電子メールシステムであり, およそ300のサイトをつないでいる. UUCPに類似した動作をする. [RFC 1168]

Action Technologies
ネットウェアMHSとしてNovellのネットウェアにバンドルされているメッセージ処理サービスの製造業者.

AD
管理領域(Administrative Domain). 単一の組織によって運営される一連のホスト, ルータ, およびネットワーク. 管理領域内のルーティングは, 首尾一貫した技術計画に基づく. ルーティングに関しては, 管理領域は外部から結合的エンティティとして見られ, その内部構造は重要ではない. 他の管理領域からきた情報よりも, 自分自身の管理領域からきた情報の方が信頼される. [RFC1136]

ADDMD
管理ディレクトリマネージメント領域(Administration Directory Management Domain). CCITTのメンバである公共通信エンティティによって管理されるX.500のディレクトリ管理領域.

ad hoc
特定のインスタンスを意味するラテン語. コンピュータ分野では, あらかじめ計画されていなかった機能を指していう.

ADMD
管理マネージメント領域 (Administration Management Domain). X.400のメッセージ操作システムの概念. 典型的には国家がADMDである. これらはある管理権限を, 私的マネージメント領域 (PRMD)に引き渡すことができる.

Adobe Systems
PostScriptページ記述言語を定義した会社.

ADT
抽象データ型を参照.

AES
アプリケーション環境仕様を参照.

AFI
権限および形式識別子(Authority and Format Identifier). AFIの後にどんな型のアドレスが続くかを知らせるためのOSIアドレスの一部.

AFP
AppleTalkファイリングプロトコルを参照.

Agenda
時間管理に用いられるLotusのソフトウェア.

AIX
Advanced Interactive Executive. IBMバージョンのUnix.

AK TPDU
確認トランスポートプロトコルデータ単位(Acknowledge Transport Protocol Data Unit). OSIト

ランスポートサービス中のプロトコルデータ単位の型の1つ.

ALTERNET

TCP/IPに基づく商用ネットワークで，商用UUCPのUUnetサービスを運営する人たちによって運営されている.

American Wire Gauge

電線の太さを記述するための標準.

AMP

アクティブモニタ存在を参照.

ANSI

米国標準化機構を参照.

APDU

アプリケーションプロトコルデータ単位(Application Protocol Data Unit)．アプリケーション層のPDU.

API

アプリケーションプログラミングインターフェイス(Application Programming Interface)．あるサービスを実行するためにアプリケーションが呼ぶことのできる関数ライブラリのインターフェイス．[SunNet Manager]

APPC

高度プログラム間通信を参照.

Apple

AppleTalkやMacintoshの製造会社.

AppleShare

異なる種類の計算機が印刷のようなAppleTalk資源を供給あるいは利用できるようにするプログラム.

AppleTalk

Apple社のネットワークプロトコル.

AppleTalkセションプロトコル (AppleTalk Session Protocol)

AppleTalkのセション層のプロトコル.

AppleTalkトランザクションプロトコル (AppleTalk Transaction Protocol)

AppleTalkのトランスポート層のプロトコル.

AppleTalkファイリングプロトコル (AppleTalk Filing Protocol)

AppleTalkでデータを遠隔アクセスするために使われるプロトコル.

ARCnet

計算機間で2.5Mbpsのスタートポロジーをもつ LANを構成するための，Datapointやその他の会社製のデータリンクのハードウェアとソフトウェアの構成要素.

ARP

アドレス解決プロトコルを参照.

ARPA (Advance Research Projects Agency)

DARPAの古い名前.

ARPANET (Advance Research Projects Agency Network)

国防省が資金を出した軍および研究団体のネットワーク．Defence Data Networkによって置き換えられた.

ARPOSE (Specification de l'Architecture et des Protocoles OSI permettant de rendre le service le services Session)

フランスのOSIプロファイル.

AS

自律システム(Autonomous System)．共通の技術管理の下にあるルータ群.

AS/400

アプリケーションシステム/400 (Application System/400)．IBMのSystem/38とSystem/36シリーズを置き換えた中型計算機.

ASCII

米国情報交換用標準コード(American Standard Code for Information Interchange)．各文字，数字およびいくつかの制御キャラクタに対して8進の列を割り当てた標準キャラクタセットである．他の主な符号化標準としてはEBCDICがある.

ASE

アプリケーションサービス要素 (Application Service Element).

Ashton-Tate

dBaseやFrameworkなどのデータ管理製品の製造会社.

ASN.1

抽象構文記法1 (Abstract Syntax Notation One). OSIプレゼンテーション層プロトコル.

ASP

AppleTalkセションプロトコルを参照

AST

非同期システムトラップ(Asynchronous system trap)．VMSのロックマネージャで使われている概念．ASTはある特定のイベントが起きたときに通知してもらう要求である．ロックマネージャの場合，あるリソースをロックしているプロセスが，他のプロセスに対して互換性がない(incompatible)ロックをしようとするとき，通知してもらうようにASTを設定することができ

る．

Async

非同期(Asynchronous)．キャラクタを1つずつ送るデータ伝送方式．データをパケットで送るとそれぞれのクロックを再同期化する同期伝送方式と対比される．非同期という言葉はコマンドにも用いられる．例えば，ウィンドウ環境において，前のコマンドの応答を待たずに次のコマンドを送ることを指す．

AT

先進技術(Advanced technology)．IBM PC/ATおよびその他のDOSオペレーティングシステムが動くIntelの80286に基づくプロセッサを指す言葉．

ATM

非同期伝送方式 (Asynchronous Transfer Mode) あるいは自動払出機 (Automated Teller Machine)．非同期伝送方式は"高速パケット"とも呼ばれている．セルごとに帯域幅を動的に割り当てる方式である．

ATP

AppleTalkトランザクションプロトコルを参照．

AT&T

アメリカ電信電話会社(American Telephone and Telegraph)．長距離通信サービスの提供会社であり，またUnixオペレーティングシステムで知られている計算機会社でもある．

AUTH

認証サービス(authentication Service)．RFC 931で定義されたサービスである．

AWG

American Wire Gaugeを参照．

Aモード(A-mode)

非同期モード(Asynchronous Mode)．OSI仮想端末プロトコルで使用される．両方がいつでも通信できるような端末動作のモデル．

b

ビット–2進桁．計算機に格納できる最小のデータ量．[RFC 1177]

B

バイト．1キャラクタの情報量であり，通常は8ビット長．[RFC 1177]

Banyan

VINESというパソコン用ネットワークの製造会社．

BASIC

Beginner's All-purpose Symbolic Instruction Code. プログラミング言語の1つ．

BAT

2進絶対時間を参照．

bayonet nut connector

10BASE5*の同軸ケーブルに使われるコネクタである．bayonetという言葉は，コネクタが接続をロックするために滑り込んでからひねられることを指している．

(*訳注：10BASE2 が正しい)

BBN

Bolt,Beranek,and Newman,Inc. ARPANET，そしてその後Internetのコアゲートウェイシステムである CSNET協調情報センター (Coordination and Information Centre; CIC)とNSFnetネットワークサービスセンター(NSFnet Network Service Center; NNSC)の開発，運用，監視を担当している，マサチューセッツ州ケンブリッジにある会社．

BBOARD

掲示板 (bulletin board).

Beepy

Sun Microsystemsの人員に関するプロジェクト (liveware project) であるBrian Pawlowskiのコード名．

BER

基本符号化規則(Basic Encoding Rules).

BIOS

基本入出力システム(basic input/output system). データにアクセスするためのMS-DOSのサブルーチンライブラリ．ディスクの物理的な特性の差異をアプリケーションから見えなくする．

Bisynch

同期式プロトコルの1つで，かつてIBMの遠隔処理環境で用いられた．BSCも参照．

BITNET

Because It's Time Network．およそ2500のホストコンピュータから構成されるネットワークで，ホストは主に大学であり多くの国にまたがっている．EDUCOMが管理しており，運営上のサポートと情報サービスを行っている．米国とメキシコのBITNET，カナダのNETNORTH，それにヨーロッパのEARNが主要なネットワークである．ほかに日本にAsiaNetがあり，南アメリカとも接続されている．CRENを参照．[RFC1177]

BLICN
物理的ネットワークの種類の1つ.

BNC
Bayonet nut connectorを見よ.

BNF
Backus-Naur記法(Backus-Naur Form). 言語の文法を記述する方法の1つ.

BOOTP
Bootstrap Protocol. ディスクをもたないノードを立ち上げるためのプロトコル. RFC951に述べられている.

BOS
Binary object sequence. PostScriptインタプリタにアクセスするための最も下のレベル. より高度なレベルとして, PostScriptプログラミングライブラリ呼出しおよびASCII文字によるインターフェイスが用意されている.

bps
bits per second. モデムや電話回線, その他のデータ通信装置の伝送速度の単位. 1秒間に送れるビットの数.

BRA
ブロードキャスト近接ルータ(Broadcast router adjacency)を見よ.

brouter
Bridge/router. ネットワークレベル, データリンクレベルの両方でネットワーク間のメッセージ受渡しを行う装置.

BSC
Bisynchronous. Bisynchを見よ.

BSD
Berkeley Software Distribution. カリフォルニア大学バークレー校で開発されたUnixのバージョンをいうときに使われる用語. "4.3BSD Unix"というように用いられる. [RFC 1177]

BSI
英国規格協会 (British Standard Institute).

Btree
2進木(binary tree)を見よ.

Btrieve
サーバからのデータを取り出すのに2進木アクセス法(Btree access method)を用いることのできるNovellのプログラム.

C
プログラミング言語の1つ. Unixオペレーティングシステムの動作しているコンピュータでよく用いられる.

CA
証明機構(certification authority)を見よ.

CAB
カーネギーメロン大学の高速光ファイバLAN実験システムNectarで用いられた, 通信専用プロセッサボード. 各ホストはこのプロセッサボートを通してネットワークのクロスバースイッチと接続されている. CABとユーザプロセスは共有メモリを用いて通信する. CABがすべてのプロトコルの処理を行うため, ホストの負担は軽くなる. [RFC1152]

CACM
Communications of the ACM. 技術雑誌.

CAD/CAM
計算機支援設計/計算機支援生産(Computer aided design/computer aided manufacture). 開発環境を自動化するためのソフトウェア／ハードウェアの組合せ.

CASA
5つのInternetのギガビット試験ネットワークのうちの1つ. このプロジェクトには, ロスアラモス国立研究所, カリフォルニア工科大学, サンディエゴスーパーコンピュータ研究所を結ぶWANが含まれている.

CASE
Computer-Aided Software Engineeringを見よ.

catenet
ホストが異なる性質をもったネットワークに接続されており, さらにそれらどうしがゲートウェイ(ルータ)で接続されているようなネットワーク. Internetがcatenetの一例である. [RFC1177]

CATV
Community Antenna TV. 広帯域ネットワークで用いられるケーブルの一種.

CBC
Cypher Block Chaining. DES暗号化法のモードの1つ.

CCA
Conceptual Communication Area. ISOの仮想端末サービスの一部. 2つのアプリケーション間で, 仮想端末が同じようにみえるような抽象的なエリアのこと.

CCIRN
Coordinating Committee for Intercontinental Research Networks. 合州国のFNCと, 北米およびヨーロッパにおけるそれに対応する機関から構

成される委員会．FNCとEuropean Association of Research Networks(RARE)の会長が委員長をつとめ，北米とヨーロッパの研究用ネットワークの計画の協調をとる．[RFC160]

CCITT
Comité Consultatif International Télégraphique et Téléphonique(Consultative Committee for International Telephone and Telegraph, 国際電信電話諮問委員会)．International Telecommunications Unionによって運営される国際標準機構．

CCR
Commitment, concurrency, and recovery．OSIのCASEサービスの1つで，複数のユーザによる複数ノードにまたがるデータのアクセスを調整する．

CCS
Common communications support．IBMのSystem Application Architecture(SAA)の一部．

CCTA
英国中央計算機通信局(Central Computer and Telecommunications Agency)．

CD
共通ドメイン(common domain)を見よ．

CDA
混成ドキュメントアーキテクチャ(Compound Document Architecture)を見よ．

CD-ROM
Compact Disk-read only memory．原盤をもとに製造される，読出し専用の光ディスク．読出し専用データベースなどに用いられる．

CDT
Credit．OSIのトランスポートサービスでフロー制御に用いられるフィールド．

CEN
ヨーロッパ標準化委員会 (Comité Européen de Normalisation)．

CENELEC
ヨーロッパ電気技術標準化委員会 (Comité Européen de Normalisation Electrotechnique)．

CERFnet
California Education and Research Federation Network．

CHAR
character．

characteristic属性 (characteristic attribute)
システム管理において，管理者によって変更が可能で，それがエンティティの動作に影響を与えるような属性．

CHARGEN
Character Generator Protocol．RFC864で定義された選択的規格．

CheaperNet
ThinWire イーサネットケーブルの別名．

CHIPCOM
広帯域イーサネット装置の製造会社．

CIC
証明書完全検査(Certificate Integrity Check)．CICとは証明書の内容が変わっていないことを確認するために使われる数量である．

CICS
顧客情報管理システム(customer information control system)．典型的にはMVSオペレーティングシステムで使われる，IBMのデータ通信インターフェイス．

CIF
Caltech Intermediate Form． Line Printer(lpr)プロトコルのグラフィックス形式オプション．

CIGOS
Canadian Interest Group for Open Systems．

C-ISL
C 入力仕様言語(C Input Specification Language)参照．

CIバス(CI bus)
コンピュータルーム相互接続(computer-room interconnect)．VAX Cluster構成で用いられる70-Mbpsのバスとコントローラのこと．トランスポートメカニズムとして10-Mbpsのイーサネットを使うLocal Area VAX Clusters と対比されている．

CLNL
コネクションレスネットワーク層(connectionless network layer)．OSIネットワーク層．

CLNS
コネクションレスネットワークサービス(connectionless network service)．OSIネットワーク層の2つのオプションの1つ．CONS も参照せよ．

Clusters
VMSノードに対するDECのアーキテクチャ．Clusterによってのみ，複数のシステムが共通のセキュリティ/管理ドメインをもち，同じディスクドライブにトランスペアレントなアクセスをすることができる．VAX Cluster 参照．

CMIP
共通管理情報プロトコル(Common Management Information Protocol). OSI ネットワーク管理プロトコルで，機能的にはSNMPと同様.

CMISE
共通管理情報サービス要素(Common Management Information Service Element). 基本的な管理サービスを提供しているCMIPサービス要素. イベントの報告や管理データの取出しにおいて，CMISEは確認を行うサービスと確認を行わないサービスの両方を提供している. [RFC 1095]

CMOT
TCP/IP上のCMIP (CMIP on TCP/IP). OSIではなくTCP/IP上のOSI CMIPプロトコルの実装. 現行のネットワークにおいて，OSIのプロトコルスタックすべてを実装することなく，CMIPの強化された機能を利用できる.

CMS
コード管理システム(Code Management System) もしくは 対話的モニタシステム(Conversational Monitor System). コード管理システムはDECのソフトウェア製品で，VMS環境でプログラム開発用のライブラリとして用いられる. 対話的モニタシステムはIBMのVM/CMSオペレティーングシステムにおけるユーザインターフェイスである.

CMT
コネクション管理(Connection Management). ISOのFDDI規格で用いられている省略形.

CMU
カーネギーメロン大学(Carnegie Mellon University).

CN
共通ネーム(Common Name). X.400のアドレスに対する省略形.

CNRI
Corporation for National Research Initiatives

COBOL
Common business-oriented language. 最も早い時期に規格化されたコンピュータ言語の1つ. CODASYL も参照.

CODASYL
データシステム言語協議会(Conference on Data Systems Languages). COBOLおよびデータ管理のネットワークモデルを用いるデータベースのCODASYL規格をもたらした.

Communication and Control Services
APIに関するNASの概念で，アプリケーションがネットワーク上の通信サービスにアクセスできるようにするもの.

config.sys
MS-DOSの起動時に読み込まれるファイルで，付加的なデバイスドライバ名(例えば，拡張メモリやスキャナなどの周辺装置のためのドライバなど)を含んでいる.

CONS
コネクション型ネットワークサービス(Connection Oriented Network Service).

CONTENT
Content Type Header Field. RFC1049で定義された推奨規格.

COS
Corporation for Open Systems.

COSAC
Canadian Open Systems Application Criteria.

CPE
Customer Premises Equipment.

CPP
ケーブルパッチパネル 参照.

CPSR
Computer Professionals for Social Responsibility

CPU
中央処理装置(Central Processing Unit). コンピュータのコンピュータたる部分.

CR
コネクション要求(Connection Request). OSI トランスポートサービスにおけるプロトコルデータ単位の型(protocol data unit type).

CRC
巡回冗長性検査(cyclic redundancy check) 参照.

CREN
the Corporation for Research and Educational Networking. BITNET と CSNET とが最近合併して CREN がつくられた. [RFC1177]

CRLF
Carriage Return/Line Feed. 復帰/改行.

CRMA
Cyclic-Reservation Multiple-Access. 超高速ネットワークで用いられるアクセス方式.

CRT
cathode ray tube. ブラウン管.

CS
チェックサム(checksum) あるいは コンピュータ

サイエンス(Computer Science)．チェックサムは伝送されたデータの正しさを確認するための方法の1つ．

CSA

クライアントサービスエージェント(client service agent)．MAP(Manufacturing Automation Protocol)ディレクトリサービスの一部．

CSMA/CD

キャリア監視・複数アクセス/衝突検出参照．

CSNET

Computer+Science Network．コンピュータサイエンスの研究を行っている機関のための大規模なデータ通信ネットワーク．独自のものを含む数多くのプロトコルが使われている．CSNETのサイトには大学，研究所，企業が含まれている．[RFC1177]

CTERM

Communications Terminal Protocol．DNA(Digital Network Architecture)第6層での，仮想端末サービスの部分．CTERMサービスに代わるものとしてLAT(Local Area Transport Architecture)がある．

CTM

現変換行列(current transformation matrix)．PostScript画像システムで，ユーザ空間とデバイス空間との変換を表すのに用いられる用語．

CUA

共通ユーザアクセス(common user access)．IBMのSystem Application Architectureの共通ユーザインターフェイス要素．類似の概念はXeroxのOpen Look規格やDECwindows環境にも存在する．

CUDF

Call User Data Field X.25ヘッダの中のフィールド．

CUG

クローズドユーザグループ(Closed User Group)参照．

Cullinet

IDMSデータベースパッケージを販売しているソフトウェア会社．

Culprit

IBM環境用のCullinet社のソフトウェアパッケージで，IMSのような別のDBMSからデータを取り出すために使われる．

C 入力仕様言語(C Input Specification Language)

NetWise RPC仕様を書くために使われる分散環境(distributed environment)用に文法が拡張されたANSI C．[Netwise RPC Tool]

DA

宛先アドレス (Destination Address)

DAP

Division Advisory Panel または Data Access Protocol または Directory Access Protocol．Division Advisory Panel は National Science Foundation のネットワーク・通信研究インフラストラクチャ分科会のこと．

DARPA

Defense Advanced Research Projects Agency．国防省の機関で，ARPANET，バークレー版Unix，TCP/IP などの多数のコンピュータプロジェクトに出資している．

DAS

Dual Attachment Station を参照．

DATAPAC

カナダ公共パケット交換網．

DAYTIME

Daytimeプロトコル．RFC867に定義された選択的プロトコル．

DB2

関係モデルとSQL問合せ言語に基づくIBMのデータベースパッケージ．

DBMS

データベース管理システム(Database Management System)．データを集中管理するソフトウェアで，同時に複数のユーザを扱い，アクセスを制御し，SQLのような高級なデータ処理言語を利用することができるようにするもの．

DC

切断確認(Disconnect Confirm)．OSIトランスポートサービスで定められているプロトコルのデータ単位の型のひとつ．

DCA

ドキュメント内容アーキテクチャ(Document Content Architecture)または国防省通信局(Defense Communications Agency)．前者はIBMのアーキテクチャの1つで機能的にDECのCompound Document Architecture (CDA)に似ている．後者は国防省データネットワーク(Defense Data Network)を担当する機関．

DCC

データ国別コード(data country code)参照．

DCE

データ通信装置 (data circuit-terminating

equipment). X.25ネットワークの用語で，ネットワークの末端にあり，呼出しを受けたり発したりする装置のこと．DCEはユーザと応対するDTEと接続される．

DCL
ディジタルコマンド言語(digital command language). VMSオペレーティングシステムのユーザインターフェイス．UNIXオペレーティングシステムのCシェルにあたるもの．

DD
ドメイン内定義(Domain Defined). X.400アドレスが局所的に定義されていることを表す属性．

DDN
国防省データネットワーク(Defense Data Network). 国防省と，国防省との契約者で利用するネットワークで，TCP/IPとX.25ネットワーキングプロトコルを使っている．

DDN NIC
スタンフォード研究所(SRI International)にあるネットワーク情報センター．RFCやインターネット草案(Internet drafts)の主たる保管場所であるが，他にもサービスを提供している．[RFC 1177].

DDP
データグラム転送プロトコル (Datagram Delivery Protocol). AppleTalkのネットワークレベルのプロトコル．

DDS
分散ディレクトリサービス(Distributed Directory Service)を参照．

DEC
ディジタルイクイップメントコーポレーション(Digital Equipment Corporation).

DECconnect
設備(facilities)の接続をするために用いられるDECのケーブル接続アーキテクチャ．

DECmcc
DEC管理制御センター(DEC Management Control Center). EMA管理アーキテクチャを実装したDECのソフトウェア．

DECnet
DECがディジタルネットワークアーキテクチャを実装したものを，他のベンダによる実装に対していう．

DECwindows
MITのXウィンドウシステムをDECが実装したもの．

DEK
データ暗号化鍵(Data Encription Key). メッセージテキストの暗号化(アルゴリズムの集合の中からある程度の選択が可能)やメッセージの完全性検査(message integrity check: MIC)の量を計算するために使われる．DEKは転送されるメッセージのそれぞれに対して生成される．プライバシー強化メッセージ転送を実現するためにDEKをあらかじめ配っておくことは必要ない．[RFC 1113]

DELNI
DECローカルネットワークインターコネクト(Local Network Interconnect).「イーサネットの缶詰」. DECの製造したマルチポートトランシーバ．

Delta T
時間差．ネットワーク上の連続する2つのパケットの間の時間経過を表すSnifferアナライザの表示．あるアンカーパケットが送られてからの経過時間である相対時間(relative time)に対比される．

DES
データ暗号化標準(Data Encription Standard). 暗号化手法のひとつ．

DF
Don't fragmentの略．パケットを分割してはならないことを示すIPフラグ．

DFS
分散ファイルサービス(あるいはシステム)(Distributed File Service). ネットワークファイルシステム(NFS)と類似のDECの製品．DFS， NFSどちらも他のホストのファイルがワークステーションに直接マウントされているかのように見せることができ，ディスクレスノードが実現できる．DFSはDNSネームサーバを使っている．

DIB
ディレクトリ情報ベース (Directory Information Base)を参照．

Digital Network Architecture
DECのネットワーキングアーキテクチャ．DECnetはこれを実装したものである．

DIS
Draft Information Standard. ISOの過程で正式な国際標準になる前の段階．この段階では標準は技術的には正しいとみなされ，小さな訂正しか行われないと期待される．

DISC
接続切断(Disconnect).

DISCARD
Discardプロトコル．RFC 863で定義される選択的プロトコル．

DISOSS
分散オフィス支援システム．文書の分散図書室として働くIBMの製品．DCAも参照せよ．

distinguished name
X.500の概念で，Directory Information Treeの中の位置に基づいて付けられるオブジェクトの一意名．

DIT
ディレクトリ情報木を参照．

DL
配布リスト(Distribution List)．X.400アドレスの代わりに用いられる省略形．

DLAL
2文字頭字語一覧 (dual letter acronym listing) の略．MLALも参照．

DLC
データリンク制御(Data Link Control)．データリンク層の情報を表すSniffer解析プログラムでの用語．

DMA
直接メモリアクセス(direct memory access)の略．コンピュータ上のデバイスがCPUに割り込むことなく主記憶にアクセスするもの．

DMD
ディレクトリ管理ドメイン(Directory Management Domain)を参照．

DMF-MAIL
Digest Message Format for Mailの略．RFC 1153の定義する選択的Internetプロトコル(実装の有無は自由)．

DML
データ操作言語(data manipulation language)．データベースシステムにおいて，SQLのようにデータを取り出したり，操作するために用いられる言語のこと．

DNA
Digital Network Architectureを参照．

DNA名前づけサービス(DNA Naming Service)
DNA Phase Vで多用される分散型名前づけサービス．ドメイン名前システムと区別するためにDNANSと呼ばれる．

DNS
DNA名前づけサービスまたはドメイン名前システムを参照．

DOS
ディジタルオペレーティングシステム(Digital Operating System)の略*．マイクロソフト社のIBM/PC用のオペレーティングシステム．
(*訳注：正しくはディスクオペレーティングシステム．)

DOS/VSE
ディジタルオペレーティングシステム/仮想記憶拡張版(Digital Operating System/Virtual Storage Extended)．IBM の370アーキテクチャメインフレーム用のオペレーティングシステム．

dpi
1インチ当りのドット数(dots per inch)．プリンタやスキャナの解像度の単位．

DR
切断要求(Disconnect Request)．OSIトランスポートサービスのプロトコルデータ単位の型．

Draft Standard Protocol
Internet標準の分類．IABはこのプロトコルを標準プロトコルの可能性として活発に検討している．徹底的かつ広範囲な試験や意見が求められている．意見や試験の結果はIABに提出するべきである．標準プロトコルになる前に，Draft Standard Protocolに変更が加えられる可能性がある．[RFC 1140]

DRP
Digital Routing Protocol.

DSA
ディレクトリシステムエージェント(Directory System Agent)を参照．

DSAP
宛先サービスアクセスポイント(Destination service access point)．サービスの宛先ユーザのアドレス．自ホストのデータリンクモジュールの見地からは，リモートIPXプロセスはDSAPと見なされる．

DSE
Directory Services Element. X.500サービス提供者．

DSP
ドメイン固有部分(Domain Specific Part)を参照．

DST
宛先アドレス (Destination address)．Sniffer Net-

work Analyzerの略語.

DTE

データ終端装置(Data terminal equipment)参照.

DU

Data unit. OSIファイル転送，アクセスおよび管理(FTAM)のプロトコルの一部.

DUA

ディレクトリユーザエージェント(Directory User Agent)参照.

Dual Attachment Station

1次と2次の光ファイバケーブルの両者に接続されるノードを示すFDDI用語(コンセントレータ経由でリングに接続されるノードと対比される).

DVI

Device Independent. TeXテキスト処理システムで使用される中間出力形式.

E.163

公衆電話交換網におけるCCITTの番号計画.

E.164

ISDN環境における番号付けのCCITT標準.

EA

Expedited Acknowledge. OSIトランスポートサービスのプロトコルデータ単位の型.

EARN

European Academic Research Network.

Easynet

DECの内部通信網.

EB

Encapsulation Boundary. カプセル化されたメッセージの終わりを示す取決め．ダッシュからなる行であることが多い．RFC934に記述されている.

EBCDIC

Extended Binary Coded Decimal Interchange Code. IBM環境で用いられる文字コード方式．ASCIIも参照せよ.

EBNF

Extended Backus Naur Form. RFC 822の2節で定義されるBackus Naur Formの拡張バージョン.

ECB

Electronic Codebook. DES暗号化方式のモード.

Echo

XNSにおける保守プロトコルの1つ．ネットワークを横断して情報を送り返させることにより，コネクションを検査するのに用いられる．また，TCPオプションの1つで「エコーリプライ」(echo reply)のオプションを用いることによって，遠隔サイトが受信したデータを送り返させる．往復転送時間(RTT)を測るのに用いられる.

ECL

End Communication Layerの略.

ECMA

European Computer Manufacturers Association. 欧州における主要な標準機関.

ED

Expedited Data. OSIトランスポートサービスのプロトコルデータ単位の型.

EDIFACT

経営，商業および貿易用のEDI．新しい国際電子データ交換(EDI)標準.

EGF

外部ゲートウェイプロトコル(External Gateway Protocol)．ネットワークを相互接続するルータやゲートウェイに経路情報を分配するプロトコル．[RFC 1177]

EHF-MAIL

Encoding Header Field for Mail．RFC1154で定義される選択的(Elective) Internetプロトコル.

EI

エンティティ識別子(Entity Identifier)．プライバシー強化メール(Privacy Enhanced Mail)においてあるメッセージや証明書が適用されるエンティティ(ユーザ)を表すサブフィールド.

EIA

Electronic Industries Association.

EIA-423

シリアル装置の標準インターフェイス定義.

EMA

Enterprise Management Architecture参照.

EMACS

MITで開発された，LISPプログラミング言語に基づく拡張可能なエディタ.

Email

電子メール (Electronic mail)．ユーザ間のメッセージの交換を可能にするソフトウェアおよびネットワーク.

EMUG

European MAP Users Group.

Enterprise Management Architecture

多数のディスプレイやプロトコルと動作することのできるDECのネットワーク管理ユーザイン

ターフェイスアーキテクチャ．DEC-mccはこのアーキテクチャの実装である．

Entry文 (Entry statement)

指定の状態機械用に生成されるソースコードにカスタム命令を付加するように，Netwise RPCコンパイラに指示するC言語の拡張．[Netwise RPC Tool]

EOF

End of file. ファイルシステムに対してファイルの終わりに着いたことを示す記号．

EON

Experimental OSI Network. Internet上のIPをサブネットワークとして用いる，試験的なOSIインターネットワークの提案の名前．[RFC 1070]

EON-UDP

Internet上のUDP/IPをサブネットワークとして用いる，試験的なOSIインターネットワークの提案の名前．[RFC 1070]

EOR

記録の終わり(End of record).

EPHOS

European Procurement Handbook for Open Systems.

EPS

Encapsulated PostScript．EPSファイル形式には2つの側面がある．1つは，PostScriptプログラムと低解像度ビットイメージが同じファイルに存在することを可能にする特別のファイル表現形式であり，もう1つはPostScriptコード自身を管理する指針である．このファイル形式の主要な目的としては，通常の方法によって文書整形システムに含まれる図の準備を容易にすることである．[Adobe SystemsのGlen ReidによるCompuserveへの投稿]

EPSF

EPS File Format.

ER

誤り(Error)．OSIトランスポートサービスにおけるプロトコルデータ単位の型．

Error

ネットワーク中に誤りを報告するのに用いられるXNSプロトコル．

ES

OSIにより定義されるEnd system：OSIネットワーク層のサービスをトランスポート層に提供するOSIネットワーク層エンティティ．[RFC 1070]

ESC

Escape.

ESH PDU

End System Hello PDU.OSIネットワークサービス定義(OSI Network Service Definition)による．

ESIS

End System to Intermediate System. ISO 9542に定義されているプロトコルで，同じサブネットワーク上にあるエンドシステムと中間システムが通信することを可能にする．

EST

東部標準時間(Eastern Standard Time).

Ethertype

イーサネットバージョン2.0において，ユーザの種類(例えばDECnet，NetWare，TCP/IPなど)を示すフィールド．

EUnet

European Unix Network. もともとはUUCPに基づくネットワーク．

EurOSInet

European OSI Promotional Group. 欧州におけるOSIの相互運用性(Interoperability)試験ネットワーク．

EVD

イベントディスパッチャ(event dispatcher)参照．

EWOS

European Workshop for Open Systems.

EXEC

Executable.

Exit文 (Exit statement)

指定された状態機械に入った後に実行されるように，クライアントスタブ，サーバスタブ，ディスパッチャ用に生成されるソースコードにカスタム命令を付加するように，Netwise RPCコンパイラに指示するC言語の拡張．[Netwise RPC Tool]

F.60

テレックスサービスに関するCCITT標準．

F.69

テレックスアドレスに関するCCITT標準．

F.110

海上移動体(maritime mobile)サービスに関するCCITT標準．

F.160

国際公衆ファクシミリサービスに関するCCITT標準．

F.200

テレテックスサービスに関するCCITT標準.

F.201

インターネットワーク上のテレテックスおよびテレックスサービスに関するCCITT標準.

F.300

ビデオテックスシステムに関する一連のCCITT勧告.

F.401

公衆メッセージ処理サービスの名前づけとアドレスづけに関するCCITT標準.

F.410

公衆メッセージ転送(transfer)サービスに関するCCITT標準.

F.415

公衆配達(physical delivery)サービスとの相互通信(intercommunication)に関するCCITT標準.

F.420

公衆 interpersonal messaging service に関するCCITT標準.

F.421

X.400 interpersonal messaging service とテレックスサービスとの間の通信に関するCCITT標準.

F.422

X.400 interpersonal messaging service とテレテックスサービスとの間の通信に関するCCITT標準.

F.500

国際公衆ディレクトリサービスに関するCCITT標準.

FADU

ファイルアクセスデータ単位(File access data unit). OSI FTAMサービスでのアクセスの単位.

FAL

File Access Listener 参照.

FC

フレーム制御(frame control). パケットがMAC層管理パケット(つまり, トークン, ビーコン, AMP, またはSMP)であるかLLCデータを運んでいるパケットであるかを示すトークンリングのフィールド.

FCCSET

科学, 工学, 技術に関する米国連邦政府調整委員会(Federal Coordinating Council for Science, Engineering and Technology).

FCS

フレーム検査シーケンス(frame check sequence). CRCやチェックサムのように, データパケットの完全性を保証する機構.

FDDI

ファイバ分散データインターフェイス(Fiber Distributed Data Interface)参照.

FEC

前向きエラー訂正(Forward Error Correction). FECは大陸間横断光リンクのように物理媒体の帯域幅/遅延が大きいとき有用な方法である. FECのより簡素化した形では, 単にデータの複数のコピーを送る. これによりあまり複雑な機能を必要とせずに帯域幅を犠牲にして遅延と信頼性をよくすることができる. [RFC 1077]

FFFF

15 15 15 15. 16進の数. 16進数は16を基数とする数え方で, 10から15までの数を表すのにAからFまでの記号が使われる. FFFFはすべてのビットが立っている32ビット数と等価である.

FET

最終形式テキスト(final form text). IBMの文書内容アーキテクチャ(DCA)の一形態.

FHSIZE

ファイルハンドルの大きさ(File Handle Size).

FI.FMD

関数管理データ用関数インタプリタ(Function Interpreters for Function Management Data). IBMのSNA用プレゼンテーション層プロトコル.

FIFO

先入れ先出し(First In, First Out). 待ち行列の管理手法.

File Access Listner

DECのデータアクセスプロトコルを用いて遠隔システムにアクセスを試みるユーザによりネットワークを介して起動されるプロセス. FALは自ノード上の記録管理サービスにより起動される, 遠隔ノード上のDAPを話すプロセスである.

finger

フィンガープロトコル(Finger Protocol). RFC 742で規定されている選択的Internetプロトコル.

FIPS

米国連邦政府の情報処理標準(Federal Information Processing Standard).

FNC
米国連邦政府のネットワーク諮問機関(Federal Networking Council). 合衆国(United States)政府のネットワークのための調整グループ.

FOO
例によく用いられる変数. 軍隊用語FOOBAR(F*d Up Beyond All Recognition)から派生した.

Form Interface Management System
書式管理のためのISO標準草案.

FRICC
米国連邦政府の研究用インターネット調整委員会(Federal Research Internet Coodinating Committee). 米国連邦政府のネットワーク諮問機関に置き換えられて以降は非公式なグループ.

FS
フレーム状態(frame status). パケット上のアドレスがステーション(station)により認識されたかどうか, データがコピーされたかどうかを示すトークンリングのフィールド.

FSM
有限状態機械(Finite State Machine).

FTAM
ファイル転送, アクセス, および管理(File Transfer,Access, and Management). 外部(foreign)システム上の仮想ファイル記憶へのアクセスを提供するOSIアプリケーション層サービス. DNA DAPプロトコルと似た目的をもつ.

FTP
ファイル転送プロトコル(File Transfer Protocol). 計算機間でファイルを転送するためのInternet標準の高位レベルプロトコル. [RFC 1177]

FYI
ご参考までに(For your information).

Gandalf
データ交換機を含む通信装置の製造業者.

GB
ギガビットバックボーン(Gigabit backbone)またはギガバイト(gigabyte). ギガビットバックボーンはInternetバックボーンの速度を1Gbpsに上げる試みである. ギガバイトは十億バイトのデータである.

Gb
ギガビット(Gigabit). 2^{30}ビットの情報.

Gbyte
ギガバイト(Gigabyte). 十億バイトのデータ.

GFLOPS
1秒間当り十億回の浮動小数点演算(Billion Floating Operations Per Second).

GGP
ゲートウェイゲートウェイプロトコル(Gateway Gateway Protocol). RFC 823に定義されている旧式のプロトコル.

GID
グループ識別子(Group identification).

GKS
グラフィカルカーネルシステム(Graphical Kernel System)参照.

GKS-3D
3次元のためのグラフィカルカーネルシステム(Graphical Kernel System for Theree Dimensions). GKS標準に対する3D拡張.

GM
ジェネラルモーターズ社(General Motors). 製造自動化プロトコル(MAP)の主要な支持者. 車も作っている.

GMD
Gesellschaft für Mathematik und Datenverarbeitung.

GNA
グローバルネットワークアドレッシング(Global Network Addressing). 32ビット完全アドレスに加え, 領域番号とネットワーク番号が存在することを示すHYPERchannelヘッダ中のビット.

GNU
ヌーはUnixではない(Gnu's Not Unix). Free Software Foundationにより開発されたUnix互換なソフトウェア. WGNUはColorado州のBoulderにある公衆ラジオ局.

GOSIP
政府のOSIプロトコル(Government OSI Protocols). 国際OSI標準の合衆国(United States)政府版.

GPO
政府の印刷所(Government Printing Office).

GQ
世代修飾語(Generation Qualifier). "Junior"のような修飾語を示すためにX.400のアドレスで用いられる省略形.

GRPC
一般化されたRPC(Generalized RPC). コード,

データ，実行が，手続き起動者(initiator)と同じ計算機上であることも離れていることもあるRPC機構．

HASP

ヒューストン自動スプールプログラム(Houston Automatic Spooling Program)．IBMの装置上の遠隔ジョブエントリ機能の独自の実装の1つ．Bisynchと共にRJE機能を表すものとしていまだに使われている．

HDTV

高解像度テレビ(High Definition Television)．

HELLO

内部ゲートウェイプロトコルの1つ．

HEMS

High-Level Entity Management Systems．RFC 1024で定義されているネットワーク管理とMIBに関する初期の定義．

Hierarchical Storage Controller

CIバスを用いるクラスタで使用されるスタンドアロンのディスクおよびテープコントローラ．HSCは実際には大容量記憶コントローラ用に最適化されたPDPコンピュータである．

HIPPI

High Performance Parallel Interface. コンピュータのバスを800から1600Mbpsの速さで比較的短距離延長するための最近のANSI規格．HIPPIはコンピュータ室でスーパーコンピュータをルータやフレームバッファ，大容量周辺記憶，他のコンピュータなどに接続するのにしばしば用いられる．

Hooks文(Hooks statement)

Netwise RPCコンパイラに，クライアントスタブやサーバスタブ，ディスパッチャの状態機械用に生成するソースコードの中にカスタム命令を付け加えるように指示する機能をもった，C言語の拡張．[Netwise RPC Tool]

HSC

Hierarchical Storage Controllerを参照．

HSI

高速シリアルインターフェイス(High-Speed Serial Interface)．T1速度(1.544Mbps)まで動作することのできるSunサーバ用の通信コントローラ．MCPも参照せよ．

HT

水平タブ(Horizontal Tab)．

HYPER integer

64ビットの整数．

HYPERchannel

Network Systems Corporation社が製造しているデータリンク．スーパーコンピュータセンターでしばしば用いられる．

I.120

CCITTによるISDNの記述．

I.230

ISDN加入者サービスの記述．

I.240

ISDN遠距離サービスの記述．

IA

Issuing Authority．プライバシ強化メールにおける概念で，ある証明書の完全性を保証しているグループを表す．

IAB

Internet Activities Board．Internetの設計，技術，管理に関する調整委員会．[RFC 1177]

IANA

Internet Assigned Numbers Authority.

IBM

International Business Machines Corporation.

IBM 8228 Multistation Access Unit

IBMが販売しているMAU．

IBM PC LAN

ブロードバンドを用いるIBM PCのネットワーク．

IBMトークンリング(Token-Ring)

IBMのトークンリングネットワーク．

IBMトークンリングアダプタ(Token-Ring Adapter)

IBMの販売しているトークンリングコントローラカード．

ICCB

Internet Configuration Control Board．Internet Activities Boardによって置き換えられた非公式なグループ．

ICD

International Code Designatorを参照．

ICMP

Internet制御メッセージプロトコル(Internet Control Message Protocol)．TCP/IPのIP層で経路制御メッセージを交換するのに用いられているプロトコル．

IDI

Initial domain identifier．OSIアドレスの一部．権限および形式識別子の後に来る．例えば，電話のアドレスにおけるIDIは国のコードといえよう．

IDMS
IBMシステム用のCullinet社のデータベース管理システム.

IDP
インターネットワークデータグラムプロトコルを参照.

IEC
Inter-exchange carrier.

IEEE
アメリカ電子電気学会(Institute of Electronic and Electrical Engineers). 合衆国における中心的な標準作成機関で, ローカルエリアネットワークに関する802規格を担当している.

IEEE 488
リアルタイムデータ収集のためのIEEE規格.

IEN
Internet Engineering Note. RFCシリーズの前身.

IESG
Internet Engineering Task Force Steering Group. IETFの理事会.

IETF
Internet Engineering Task Force. Internetで用いる新しい技術の開発に協力するボランティアグループ.

IGMP
Internet Group Multicast Protocol. RFC 1112に定義されている推奨Internetプロトコル.

IGP
Interior Gateway Protocol. 1つの管理ドメインの中で用いられるRIPなどのプロトコル. 管理ドメインの間で情報を交換するのに用いられるEGPと対比される.

IHL
Internet Header Length. Internet(IP)ヘッダの長さを表す.

IK
Interchange Key. データ交換鍵(DEK)を暗号化するのに用いられる鍵. プライバシ強化メールではinterchange keyは公開鍵暗号方式に基づいている.

IMAP2
Interactive Mail Access Protocol. RFC 1064に定義されている限定使用プロトコル.

IMS
情報管理システム(Information Management System). 階層データ管理モデルに基づくIBMのデータベース管理ソフトウェア.

Ingres
よく用いられているデータベース管理システムで, いろいろなオペレーティングシステムの上で動く. この名前は有名な19世紀のフランスの画家の名前に由来している.

INOC
Internetネットワーク運用センター(Internet Network Operation Center). Internetのいくつかの管理ドメインの管理をしているセンター.

INTAP
Interoperability Technology Association for Information Processing. 日本のグループ.

Intel
IBM PCで用いられている80x86マイクロプロセッサの製造会社.

interchange key
2箇所以上で共有されている暗号化の鍵.

Interleaf
VAXstationやSunワークステーションの上で動くデスクトップ・パブリッシング・プログラム.

International Code Designator
ISOで管理されている4桁の組織ごとの一意ID(地理的位置に基づくデータ国別コードと対比される).

International Telegraph and Telephone Committee
CCITTの英語名.

Internet
同一の名前空間を共有し, TCP/IPプロトコルを用いるネットワークの集まり. Internetには少なくとも400のネットワークが接続されている. Internetは, 互いの間に経路をもつすべてのネットワークを指すインターネット(小文字で表記される)と混同すべきでない. Internetのすべての構成員はTCP/IPを用いている.

Internet自律システム (Internet Autonomous System)
複数のゲートウェイからなる自律システム. これらのゲートウェイはそのシステムに属するゲートウェイのみを介してお互いに接続が可能でなければならない. これらのゲートウェイは, IGP(Interior gateway protocol)を用いて協調して経路データベースを管理する.

Internetプロトコル (Internet Protocol)
Internetのネットワーク層のプロトコル. RFC791で定義されているデータグラムプロト

コルである．[RFC1177]

Interpress
XNSで用いられる，レーザプリンタ用のページ記述言語．

I/O
入出力(input/output)の項を見よ．

IOCTL
I/O contorol．デバイスを制御するためのUnixのファンクションコール．

IOP
入出力プロセッサ(input/output processor)．Unixを用いたミニコンメーカでは，I/Oサブシステムをこう呼ぶ．

IP
Internetプロトコル (Internet Protocol)の項を見よ．

IP-ARC
Internet Protocol on ARCnet．RFC1051で定義されている選択的Internetプロトコル．

IPC
プロセス間通信(Interprocess Communication)の項を見よ．

IPDS
Intelligent Printer Data Stream．IBMのSAA(System Application Architecture)の一部．

IP-FDDI
FDDI上のIPの伝達．RFC1103で定義されている選択的Internetプロトコル．

IP-HC
Internet Protocol on HYPERchannel．RFC1044で定義されている選択的Internetプロトコル．

IPM
Interpersonal Message．X.400プロトコルの一部で，メッセージのヘッダの形式を定義したもの．MTS(Message transfer service)が，メッセージを入れる封筒の形式を定めている．

IPMS
Interpersonal Messaging System．2つのユーザエージェントが，情報を交換するのに用いるプロトコル．

IPN
Interpersonal Notification．X.400のメッセージタイプの1つで，受領の情報などを交換するのに用いられる．

IP-SLIP
シリアル回線によるIP伝搬．RFC1055で定義されている選択的Internetプロトコル．

IRSG
Internet Research Task Force Steering Group.

IRTF
Internet Research Task Force．IRTFは，主にInternetに関するネットワーク研究者の集まり．IRFTの活動はIRSG(Internet Research Task Force Steering Group)に基づく．[RFC 1177]

IS
中間システム(Intermediate System)の項を見よ．

ISAM
Indexed sequential access method．インデックスによってランダムアクセスを行い，その後は逐次的に読み出すことのできるファイル構造．

ISDN
サービス総合ディジタル網(Integrated Services Digital Network)の項を見よ．

ISH
Intermediate System Hello.

ISH PDU
Intermediate System Hello PDU．OSIネットワークサービスの定義による．

ISIS
Intermediate System-Intermediate System.

ISM
Information security Manager．データの監視役．

ISO
International Organization for Standardizationの項を見よ．

ISODE
ISO Development Environment．PSI InternationalのMarshall Roseによって開発されたソフトウェアで，TCPに基づくネットワークでOSIサービスを可能にする．

ISPF
Interactive System Productivity Faculty．IBMの製品であり，TSOやCMS上で稼働する．ISPFは3270端末上で用いる一連のメニュー(ダイアログ)を提供し，ユーザがコマンド言語インターフェイスを介さずに，オペレーティングシステムを扱うことを可能にしている．

ISV
Independent Software Vendors．サードパーティのソフトウェア開発会社．VAR，OEM，サードパーティソフトウェア開発者などとも呼ばれる．

ITSU
Information Technology Standards Unit.

ITT
Invitation to Transmitの項を見よ．

JANET
Joint Academic NETwork．JANETはイギリスにおける主要な学術ネットワークで，およそ100の大学や研究所の約1000台のコンピュータが接続されている．JANETはInternetと類似したドメインネームシステムをもつが，ドメインネームの並べ方は逆である(すなわち，トップレベルのドメインを左側に書く)．

JCL
ジョブ制御言語(Job Control Language)．IBMメインフレームで，バッチジョブに用いられる言語．

JES
Job Entry Subsystem．IBMメインフレーム上でJCLを処理するプロセスの種類．

JNT
Joint Network Team．イギリスの学術ネットワークを管理するグループ．

JSR
Jump-subroutine instruction．サブルーチンへ制御を移行する命令．

JTM
Job transfer and manipulation．OSIの7層の規格の1つで，遠隔ジョブ投入(RJE)サービスと同様の機能をもつ．

JvNC
John von Neumann National Supercomputer Center.

Kb
Kilobit．千ビット．

KB
Kilobyte．千バイト．

kbps
Kilobit per second.
1秒当り千ビット．

kbyte
Kilobyte．千バイト．

KDC
Key distribution center．2つの団体に対称のキーを配布する．Kerberosで用いられる．

keep aliveメッセージ
ユーザ間でのトラフィックが存在しない期間に，ネットワーク回線を通して送られるメッセージ．遠隔ノードが生きていることを示す．

Kerberos
MITのAthenaプロジェクトの構成要素．Kerberosは，対称鍵暗号に基づくセキュリティシステムである．RSAの公開鍵暗号と対比される．

Kermit
コロンビア大学で開発された，よく用いられているファイル転送プロトコル．Kermitは大部分の環境で動作するので，ファイル転送の簡単な手段を提供している．

KNET
Kangaroo Network．Spartacus and Fibronicsによって作られているハードウェアとソフトウェアの製品で，IBMのメインフレームがTCP/IPプロトコルを用いて，ネットワーク経由で通信できるようにする．

LAN
Local area network．通常は，イーサネットかトークンリングのネットワークを指す．

LAP
Link access precedure．HDLCデータリンクにアクセスする手続き．X.25環境で用いられるLAPBや，ISDN環境で用いられるLAPDなどがその例である．

LAPB
Link access procedure, balanced．HDLCデータリンク規格のサブセット．

large nonmoveable cabling system
IBMの偉大な製品の1つ．small moveable cabling systemの項を見よ．

LAT
Local Area Transport の項を見よ．

latency buffer
トークンリングにおける概念．このバッファはアクティブモニタによって管理され，ネットワーク上のデータの速度の差を補償するために用いられる．

LAVC
Local Area Vax Cluster．システム通信アーキテクチャ(SCA)をCIバスの代わりにイーサネット上で動作するようにしたもの．Micro Vaxをディスクレスのノードのように用いるのに用いられる．

LBL
Lawrence Berkeley Laboratory.

LF
行送り(Line Feed).

LFN

長太パイプネットワーク(Long, fat pipe network). 大きなバンド幅と長い遅延を伴うネットワーク. その結果として確認されないビット数が大きくなる. この種のネットワークは"LFN"として知られ, elephan(t)と発音される.

LHS

左辺(Left Hand Side).

LI

Length Indicator. OSIネットワーク層のES-ISプロトコルで用いられる.

LISP

List Processing Language.

LME

Layer Management Entity.

Local Area Transport

DECが開発したプロトコルで, イーサネット上の端末サーバとホストの間の通信で用いられる.

LocalTalk

AppleTalk のために開発されたデータリンクで, 配線に通常の撚り対線を用いる.

LPP

Lightweight Presentation Protocol. RFC1085で定義されている, 簡素化されたASN.1のプレゼンテーションサービス. CMOTで使われている[RFC 1095]. LPPによって, ISOのプレゼンテーションサービスを, TCP, UDPいずれの上でも利用することができる. この方法により, TCP/IP環境でISOのネットワーク管理を行うために, ISOのプレゼンテーション, セション, トランスポートプロトコルを実装する必要がなくなる.

lpr

ラインプリンタ(Line Printer).

LSP

Link state packet. 第5フェイズのDECnetや, OSI IS-ISで交換される経路制御情報メッセージ.

LU

Logical Unit. IBMのSNAでの用語で, ネットワークを用いるソフトウェアまたはマイクロコードを指す. 例えば, 3274クラスタコントローラに接続されている端末は, そのクラスタコントローラのLU2と表記される.

LU 6.2

Logical Unit 6.2のこと. 高度プログラム間通信(Advanced Program-to-Program Communication)の項を見よ.

LWSP

ホワイトスペース.

MAC

媒体アクセス制御 (Medium Access Control), またはMacintoshの項を見よ.

Macintosh

Apple Computer社製のコンピュータ. 視覚的, 直観的なユーザインターフェイスに特徴がある.

Macintoshファインダ(Macintosh Finder)

Macintoshのオペレーティングシステムのデスクトップを管理する部分.

MAC層ブリッジ (MAC-layer bridge)

2つ以上の類似のデータリンクを, そのデータリンクサービスのユーザ(つまりネットワーク層)に透明な形で相互接続する装置.

mail-11

VMS mailで使われていた元来の電子メール経路選択プロトコル. MAILbusはより新しいメッセージ処理アーキテクチャである.

MAILbridge Server

異なるメッセージ処理システムを接続する, SoftSwitch社の一連の製品.

MAILbus

DECnet上で共通のメッセージ処理システムを提供する, DECのアーキテクチャ.

MAP

生産自動化プロトコル(Manufacturing Automation Protocol)の略. General Motorsによって出資された, OSIに関連したアプリケーション. 多数のユーザやベンダから支持されている.

MAU

マルチステーションアクセスユニット(multistation access unit)を参照*.
(*訳注:Medium Access Unit(イーサネットのトランシーバ)の意味で使われることも多いようである.)

MAXFILELEN

最大ファイル長のこと.

Mb

メガビットの略. 2^{20}ビットの情報(通常, 毎秒1メガビット=1Mbpsのようにデータ転送速度を表すのに使われる).

MB
メガバイトの略．百万バイトの情報．

Mbps
毎秒百万ビット(million-bits per second)の略．

Mbyte
メガバイトの略．百万バイトの情報．

MC68000
Motororaにより生産されているCPUチップのシリーズ．Mac IIやSun-3シリーズなど多くのUnixワークステーションの基盤となっている．

MCB
メッセージ制御ブロック(message-control block)の略．メッセージのヘッダの部分を指す．

MCC
管理制御センター(Management Control Center)の略．DECmccを参照．

MCI
長距離電話会社の1つ．

MCI Mail
商用の電子メールサービス．

MCP
マルチプロトコル通信プロセッサ(Multiprotocol Communications Processor)の略．Sunのサーバのための，中程度の速度の通信コントローラである．HSIの項も参照．

MD
DNSリソースレコードの型の1つ．MXにより使われなくなった．

MDB
管理データベース(management database)の略．すべての被管理オブジェクトの属性定義や，SunNetマネージャコンソールのためのデータ定義を集めたもの．[SunNet Manager]

MEN
管理イベント発生通知(management event notification)の略．CMIPの一部で，イベントに関する情報をネットワークを介して送るのに使われる．

MF
DNSリソースレコードの型の1つ．MXの登場により使われなくなった．

MHS
メッセージ処理システム(message handling system)の略．

MIB
管理情報ベース(Management Infomation Base)の略．被管理オブジェクトがディレクタに対して利用可能としている情報の定義を集めたもの．

MIC
メッセージ完全性検査指標(Message Integrity Check)またはメディアインターフェイスコネクタ(Media Interface Connector)の項を参照．

MicroVAX
ワークステーションや小型サーバに通常用いられている，DECのプロセッサのシリーズ．SunやHewlett-Packard/Apolloと市場において競争を繰り広げている．

Milnet
Military Networkの略．非機密の軍事生産アプリケーション(military production applications)で用いられているネットワーク．Internetの一部をなす．

MIN
最小値(minimum)のこと．

Minitel
フランスのVideotexアプリケーションのための端末．

MIPS
毎秒100万命令(million instruction per second)の略．CPUの速さの尺度．

MIR
管理情報収納場所(management information repository)の略．

MIS
管理情報システム(Management Information System)の略．ある組織内の管理者に情報を提供するデータベースシステムのこと．この言葉は組織の中で，計算を担当する部門を指すようになった．

MIT
マサチューセッツ工科大学(Massachusetts Institute of Technology)の略称．X-ウィンドウシステムはここで開発された．

MLAL
複数文字頭字語一覧(multiletter acronym listing)の略．

MMDF
マルチチャンネル・メモ分配機構(Multichannel Memo Distribution Facility)の略．Unixシステム上のメールハンドラの1つ．

MOP
メンテナンス操作プロトコル(Maintenance Operation Protocol)の項を参照．

MOTIS

メッセージ指向テキスト相互交換システム(Message Oriented Text Interchange System)の略．1988 CCITT X.400標準の正式名称．

MPR

マルチポートリピータ(multiport repeater)の略．

MPT

マルチポートトランシーバ(multiport transceiver)の略．

MS

メール区分(Mail Stop)の略．

MT

メッセージ伝搬(Message Transfer)の略．

MTA

メッセージ伝搬エージェント(message transfer agent)の項目を参照．

MTS

メッセージ伝搬サービス(message transfer service)の略．情報のエンベロープのやりとりを制御するX.400のプロトコルのこと．IPMSにエンベロープの内容が定義されている．

MTTF

平均故障寿命(Mean Time to Failure)の略．ハードウェア故障やサービス喪失が発生するまでの平均の時間．測定によって経験的に求められる場合もあるし，ひとつひとつの構成要素のMTTFから算出する場合もある．[RFC 1177]

MTTR

平均修復時間(Mean Time to Recovery)の略．故障やサービス喪失が発生してから，サービスが復旧するまでの平均の時間．通常は経験的に測定される．[RFC 1177]

MTU

最大転送単位(maximum transfer unit)の略．与えられたサブネットワーク上を転送できる最大の大きさのデータ．

MVS/TSO

多重仮想記憶/タイムシェアリングオプション(multiple virtualstorage/time sharing option)の略．MVSはIBMのオペレーティングシステムの名称である．TSOは対話型のサブシステムで，バッチ処理のために用いられるJESのようなシステムと対比される．

MX

メール交換(Mail Exchange)のこと．特定のドメインへのメールをどのホストが処理するかを示すDNS資源レコードの型．

NAK

否定応答(Negative Acknowlegement)のこと．データの未受信または誤りのあるパケットの受信に対する応答．

namedropper

ドメイン名前システム(Domain Name System)に関するメーリングリスト．

NAMESERVER

ホスト名サーバプロトコル(Host Name Server Protcol)のこと．IEN-116に定義されていた現在では用いられていないInternetプロトコル．

NAS

Network Application Supportを見よ．

NASA

アメリカ航空宇宙局(National Aeronautics and Space Administration)のこと．

NAU

Network addressable unit のこと．IBMのSNAネットワークの境界．論理ユニットや物理ユニットがNAUの例である．

NBP

名前結合プロトコル(Name Binding Protocol)を見よ．

NBS

国家標準局(National Bureau of Standards)のこと．国立標準技術研究所(National Institute for Standards and Technology)を見よ．

NAS-AS2

国家標準局，抽象構文2(National Bureau of Standards, Abstract Syntax 2)のこと．NBS(現NIST)によって開発された，ファイルディレクトリをファイルのように見えるエントリにするための抽象構文．これによって，遠隔FTAMシステムはファイルのディレクトリを要求できるようになる．

NCS

Network Computing Systemを見よ．

ND

NetDiskのこと．生(raw)のディスクをネットワークを経由してロードするためのSunのプロトコル．機能上は，DECのMOPプロトコルと似ている．今は使われていない．

NEARnet

ニューイングランド学術研究ネットワーク(New England Academic and Research Network)のこと．

NetBIOS
ネットワークアダプタ基本入出力システム(Network Adaptar Basic Input/Output System)のこと．クライアントプログラムがサーバプロセスを見つけて通信することができるようにするためのネットワークプロトコル．名前つきパイプ(Named Pipes)に似ている．

NETBLT
大量データ転送プロトコル(Bulk Data Transfer Protocol)のこと．RFC 998で定義されていた，現在では使われていないInternet高速ブロック転送プロトコル．

NETSTAT
ネットワーク統計情報収集(Network Statics)のこと．RFC 869で定義されているネットワーク監視プロトコル．

NetWare
Novell社から発売されているネットワーキング構成要素．データリンクドライバ，トランスポートプロトコル階層，ワークステーションソフトウェア，NetWareオペレーティングシステムをまとめたものである．

NetWare Core Protocols
NetWareファイルサーバによって提供されている中核サービスを利用するのに使うプロトコル．ファイルアクセス，ロック機構，プリント機構，ジョブ管理などの種々の機能を含む．

Netwise
Netwise RPC Toolの製造業者．

Network Application Support
すべてのソフトウェアが全機種のハードウェア・ソフトウェアと動作することを可能にするためのDECの標準．範囲と有効性は，IBMのSAAと似たものである．

Network Computing System
アポロ社のコンピューティングアーキテクチャ．DECのRPC機構は，NCSのRPCアーキテクチャに由来している．

Network Loadable Module
ロードされるとオペレーティングシステムの一部となる，Netware 386上のプログラム．

NeWS
Network Extensible Window Systemのこと．Sun Microsystems社から提供されている，PostScript言語と独自のウィンドウ制御プロトコルに基づくウィンドウ環境．

NEXT
Steve Jobsによって設立されたコンピュータ会社である．Steve Jobsは，以前はApple社に所属していた．

NFBLK
NFS Blockのこと．そのファイルがブロック型特殊ファイル(block special file)であることを示すファイルの属性．

NFCHR
NFS Characterのこと．そのファイルが文字型特殊ファイル(character special file)であることを示すファイル属性．

NFDIR
NFS Directoryのこと．そのファイルがディレクトリ(directory)であることを示すファイル属性．

NFLNK
NFS Linkのこと．そのファイルがシンボリックリンク(symbolic link)であることを示すファイル属性．

NFNON
NFS Named Socketのこと．そのファイルが名前つきソケット(named socket)であることを示すファイル属性．

NFREG
NFS Rgular Fileのこと．そのファイルが通常ファイル(regular file)であることを示すファイル属性．

NFS
Network File Systemのこと．あるコンピュータ上で動いているプログラムが，同じインターネット上の他のコンピュータにあるデータを，あたかも自分のマシンのディスク上にあるかのように使用できるようにするネットワークサービス．[RFC 1177]

NIC
ネットワーク情報センター(Network Information Center)のこと．スタンフォード研究所(Stanford Research Institute)内の機関で，Internetアドレスを管理している．

NICE
Network Information and Contorol Exchangeのこと．ネットワーク管理情報の交換に使用される，DECnetのプロトコル．

NIS
ネットワーク情報サービス(Network Information Services)．イエローページ(Yellow Pages)を参照

せよ．

NIST
国家標準技術局(National Institute for Standards and Technology)を見よ．

NLM
Network Loadable Moduleを見よ．

NNSC
NSF Network Service Center.

NNTF
Networking and Telecommunications Task Force.

NOC
ネットワーク運用センター(Network Operations Center)．ネットワークの維持に責任をもつ組織である．[RFC 1177]

NOOP
無操作(No Operation)のこと．

Novell
NetWareというネットワーク用ソフトウェアの製造業者．

NPDU
ネットワークプロトコルデータ単位(Network Protocol Data Unit)のこと．

NPID
ネットワーク層プロトコル識別子(Network Layer Protocol Identifier)のこと．OSIのネットワーク層のES-ISプロトコルで用いられる．

N(R)
受信順序番号(Recieve sequence number)のこと．最後に受信したパケットの順序番号を示すLLCフィールドである．

NREN
全国研究教育ネットワーク(National Research and Education Network)のこと．

NRS
名前登録機構(Name Registration Scheme)，または次受信順序(Next Recieve Sequence)のこと．Name Registration Schemeは，英国におけるドメイン名前システム(Domain Name System)の相当品である．Next Recieve Sequenceは，(応答内では)対応する要求メッセージを，(要求内では)前回の応答を，前回の応答に続いてこのエンティティから受信したことを示すVMTPのフラグ．つまり，他のメッセージを受け取らなかったということである．

N(S)
送信順序番号(Send sequence number)のこと．送出中のパケットの順序番号を示すLLCフィールドである．

NSAP
ネットワークサービスアクセスポイント(Network Service Access Point)を見よ．

NSDU
ネットワークサービスデータ単位(Network Service Data Unit)のこと．

NSF
全米科学財団(National Science Foundation)のこと．

NSFnet
全米科学財団ネットワーク(National Science Foundation Network)のこと．国全体に及ぶ高速インターネットであり，研究応用の目的で作られた．NSFnetバックボーンとNSFnet地域ネットワークからなる．これは，Internetの一部である．

NSFnet地域ネットワーク (NSFnet Regional Network)
アメリカ合州国のある地域をカバーする，NSFnetバックボーンに接続されたネットワーク．ローカルサイトは，この地域ネットワークに接続される．

NSFnetバックボーン(NSFnet Backbone)
アメリカ合州国大陸中の13サイトをつなぐネットワーク．NSFnetの中心的な構成要素である．

NSI
NASA科学Internet (NASA Science Internet)のこと．NASAの広域ネットワークである．

NSP
ネットワークサービスプロトコル(Network Service Protocol)を見よ．

NSS
NSFnetバックボーンにおける主たるルーチングノード．

NVP
Network Voice Protocol．ISI-memoで定義されている制限使用Internetプロトコル(Limited Use Internet Protocol).

NVT
ネットワーク仮想端末(Network Virtual Terminal)．Telnetサービスで定義されている仮想端末．

NWNET
NorthWestNetのこと．

NYSERnet
ニューヨーク州教育研究ネットワーク(New

York State Educational and Research Network)のこと．ニューヨークの教育研究機関をサービスしているインターネットである．また，ニューヨーク州のNSFnet地域ネットワークの役割も果たしている．

OA

Office automation．今月号のDatamation誌に載っているようなこと．

OCR

光学的文字認識(Optical Character Recognition)．

OEM

もとの部品製造者(Original equipment manufacturer)．他社の製品に埋め込まれる部品を売る会社のこと．その相手方の会社は製品の付加価値再販売者(VAR)として知られている．

OID

オブジェクトID．

OIW

OSI実装者ワークショップ(OSI Implementors Workshop)．NISTによって主催されている一連のワークショップ．

OLTP

オンライントランザクション処理(On-line Transcation Processing)．マーケティング部門であいまいにされた用語で，よくて速いものすべてを意味する．実際には，たくさんのトランザクションを迅速に実行するアプリケーションを指す．

ONC

オープンネットワークコンピューティング(Open Network Computing)参照．

O/R address

発信元と受信者のアドレス(Originator/recipient address)．正当なX.400のアドレス．

OS

オペレーティングシステム．

OS/2

Operating System/2．IBM PCに関するDOSの後継者(誰もが欲しいと言っているもののこと)．

OSF

オープンソフトウェア基金参照．

OSI

オープンシステム間相互接続参照．

OSIE

オープンシステム間相互接続環境参照．

OSI参照モデル(OSI Reference Model)

OSIの7つの階層とそれらの機能を定義したOSIの「概要」．他のネットワークを説明するのにもときどき使われる．[RFC1177]

OSPF

Open Shortest-Path First．実験的にRIPを置き換えるもの．RIPのいくつかの問題を取り上げて，Internet以外のプロトコルで，よくテストされた原理に基づいている．

OU

組織の単位(Organization Unit)．組織の下位の単位を示すX.400のアドレス属性．

PAD

パケット組立て / 分解装置 (Packet Assembler/Disassembler)参照．

PAP

プリンタアクセスプロトコル(Printer Access Protocol)参照．

PAR

Positive acknowledgement retransmit．DDCMPデータリンクプロトコルで用いられている，通信の信頼性を保証する手段．

Paradox

IBM PCでよく使われているデータベースプログラム．

PARC

Xeroxパロアルト研究センター (Xerox Palo Alto Research Center)参照．

PBR

policy-based routing．

PBX

構内交換機 (Private branch exchange)．利用者の構内に設置された電話交換機．

PC

パーソナルコンピュータ(Personal Computer)．IBMによって開発されたデスクトップコンピュータや，第三者のベンダによって開発された複製品．ときおり，PC はより一般的に，Apple Macintoshのような他のデスクトップシステムを指すのに使われることがある．

PC-DOS

パーソナルコンピュータディジタルオペレーティングシステム(Personal Computer-Digital Operating System)*．Microsoft社のオペレーティングシステムのIBM版．

(*訳注　正しくは，パーソナルコンピュータディスクオペレーティングシステム(Personal Computer-Disk Operating System)．)

PC-NFS

Personal Computer Network File System.

PD

プロトコル草案(Protocol Draft).

PDL

ページ記述言語(Page Description Language). Postscriptが PDLの一例である.

PDP

programable data processor. DECによって製作されたQバスに基づく16ビットミニコンピュータのシリーズ*.

(*訳注：初期のPDPシリーズはユニバスと呼ばれるバス規格を用いていた.)

PDS

郵便物配達システム(Postal Delivery System), または構内配線システム(Premises Distribution System). 郵便物配達システムはsnail mailとして知られている. 構内配線システムは, AT&Tの配線システムである.

PDU

プロトコルデータ単位(protocol data unit)参照.

PEP

パケット交換プロトコル(Packet Exchange Protocol)参照.

PEX

PHIGSのXへの拡張(PHIGS Extention to X)参照.

PHIGS

プログラマ用階層的対話型グラフィックシステム(Programmer's Hierarchical Interactive Graphics System)参照.

PHIGSのXへの拡張(PHIGS Extention to X)

3Dグラフィックを合わせたXウィンドウシステムへの拡張.

PHY

物理的(Physical).

PICS

Protocol Implementation Conformance Statement. プロトコル実装が仕様を満足していることを述べたもの.

PID

プロセスの識別番号(process identification number), またはプロトコルID(protocol ID). プロセス識別番号は, オペレーティングシステム上で走っている各々のプロセスを区別するために用いる. プロトコルIDは, サービスの1レベル上のユーザを示すのに用いられる.

PN

個人名(Personal Name). 個人の名前を示すX.400のアドレス属性.

POP

Post Office Protocol. RFC1081に記述されたメッセージプロトコル.

POSI

Promotion of OSI. 日本のグループ.

POSIX

Portable Operating System Interface for Computer Environment. オペレーティングシステムに対して, インターフェイスを共通にするためにIEEEが開発した標準で, これにより, アプリケーションの移植性を高めることができる.

PostScript

Apple LaserWriterのようなプリンタや, NeXTやSun Microsystemsのような会社のワークステーションで使われている, コンピュータディスプレイ上で用いられるページ記述言語. XeroxのInterpressと機能は似ている.

POTS

単なる古い電話サービス(Plain Old Telephone Service). ISDNやキャッチホン, その他の最近のサービスと対比するためのもの.

PPDU

プレゼンテーションプロトコルデータ単位(Presentation Protocol Data Unit).

ppm

1分当りのページ数(Pages per minute). レーザプリンタの速さの単位.

PRDMD

Private Directory Management Domain. 公衆のサービス提供者ではない, X.500の管理エンティティ.

PRMD

私的マネージメント領域 (Private manegement domain). X.400の管理領域. ADMD参照.

PROFS

Professional Office System. IBMのVM/CMSオペレーティングシステム用のオフィス自動化パッケージ.

Proposed Standard Protocol

Internet標準の分類. IABで将来の標準として検討される可能性のあるプロトコルの提案. いくつかのグループにより実装とテストがなされることが望ましい. プロトコルの仕様が修正される可能性がある. [RFC 1140]

proxy system
代理エージェント(proxy agent)が実行されるホスト．[SunNet Manager]

PS
Presentation Service.

PS profiles
Presentation services profiles.SNAの用語で，2つのNAUが共にサポートできるプレゼンテーション機能の受け入れ可能なサブセットを交渉するのに用いられる．

PSC
Pittsburgh Supercomputing Center.

PSI
Packet-switch interfaceまたはPerformance Systems International.Packet-switch Interfaceは，VAXをX.25ネットワークに組み込むためのDECのソフトウェアである．Performance Systems InternationalはISODE，X.500，SNMPの領域におけるネットワークサービス供給者であり，活発な参加者である．

PSN
パケット交換網(Packet-switched network)．X.25のネットワーク．

PSTN
公衆電話網(Public Switched Telephone Network).

PTR
Pointer.DNSのリソースレコード種別で，ある名前と他の名前の間の別名の機能を提供するために用いられる．アドレス(INADDR.ARPAレコードとして格納されている)とホスト名との間を指し示すためによく用いられる．

PTT
Poste Tèlèphone et Tèlègraphe.ヨーロッパのほとんどの国における通信機能の国家的な供給元．

PU
Physical unit.SNAの用語で，ネットワーク中のいろいろな種類のハードウェアを指す．3274クラスタコントローラはPUタイプ2(PU2)である．

PVC
ポリ塩化ビニル(Polyvinyl chloride)，またはパーマネントバーチャルサーキット(parmanent virtual circuit)参照．ポリ塩化ビニルはケーブル被覆の一種で，屋外には適さないが建物の中ではしばしば用いられる．

PVP
Packet video protocol.

Q.700
CCITT SS No.7の紹介．

Q.701
Signalling System No.7のメッセージ転送部分．

Q.711
Signalling System No.7の信号コネクション制御部分．

Q.721
Signalling System No.7の電話利用者部分．

Q.730
Signalling System No.7のISDN追加サービスの定義．

Q.761
Signalling System No.7のISDN利用者部分．

Q.930
レイヤ3におけるISDNユーザネットワークインターフェイスのCCITT標準．

Qバス(Q-bus)
MicroVAXやPDPコンピュータで用いられている周辺バス．

QOS
サービス品質(Quality of service)．X.25とOSIのネットワーク実装において，一連の交渉可能なパラメータ．

Quarterdeck Office Systems
DOSが動いている80386コンピュータにいくつかの異なる仕事を実行させるためのプログラムであるDesqview/386の製造業者．

QUOTE
Quote of the Day Protocol．RFC 865で定義されている選択的Internetプロトコル．

R
応答(Reply)．応答パケットに対してSniffer Analyzerが付けるラベル．Cはコマンドを意味する．

RAM
Random access memory．ダイナミックメモリであり，しばしば主記憶やコアとして知られる．

RARP
逆アドレス解決プロトコル (Reverse Address Resolution Protocol)．TCP/IPのプロトコルで，ARPの逆の機能を提供する．ディスクレスノードが最初に初期化するときに自分のInternetアドレスを見つけるために用いられる．

RCL
Revoked Certificate List.X.509セキュリティで用いられるリストで，どの証明書がもはや有効でないかを示す．

rcp
Remote copy program. ファイルのコピーを行う，Berkeley Unixで実装されたTCP/IPの上位レベルサービス．FTP参照．

RD PDU
Redirect PDU.OSIネットワークサービスの定義による．システムに対し，今後のPDUを他の中間システムに再方向づけするよう知らせるために用いられる．経路制御の簡単な形である．

RDA
Remote Data Access.異機種コンピュータ環境においてデータベースにアクセスするための国際標準．

RDATA
Resource Data. DNSリソースレコードの可変部分．

RDN
Routing Domain Number. OSIアドレス中の，Internet Assigned Numbers Authorityにより割り当てられた部分．

RDP
Reliable Data Protocol. RFC 908と1151で定義された限定使用のInternetプロトコル．

READDIR
Read Directory. ディレクトリの内容を読み出すNFSの操作．

READLINK
シンボリックリンクがどこを指すかを見い出すNFSの操作．

reference
参照されるメモリが既に割り付けられているような特別な種類のポインタ．[Netwise RPC Tool]

REJ
Reject.

reject PDU
サーバのRPCコードにより作成され，クライアントのアプリケーションプロセスに返されるPDUで，RPCコードにより検出されたエラーに関する情報が記載される．[Netwise RPC Tool]

Request for Comment
InternetのRequest for Comment文書シリーズ．RFCはInternet研究開発コミュニティの作業文書である．このシリーズのドキュメントはコンピュータ通信に関して基本的にあらゆる項目を扱っており，会合報告から標準仕様まであらゆるものがある．[RFC 1177]

resource fork
Macintoshファイルシステムにおけるファイルの第2の部分．resource forkには実行コードが含まれており，data forkにはユーザデータが含まれている．

RFC
Request for Comment参照．

RG-62
同軸ケーブルの定格．

RGB
Red, green, blue.3原色を混ぜ合わせて色を表現する方法．

RHS
右辺(Right Hand Side).

ring purge
ネットワークのデータを一掃するトークンリングパケット．ARCnetのreconfigureation burstと似た機能．

RIP
経路情報プロトコル(Routing Information Protocol)参照．

RISC
Reduced instruction set computer. 従来の方式より単純な命令セットをもっているCPUの総称．例としてIBMのPC/RT，Pyramid minicomputer，Sun SPARCワークステーションなどがある．

RJ
Reject.OSIトランスポートサービスのプロトコルデータ単位の種類．

RJ11
AT&Tにより開発された標準モジュラジャック．電話やデータ通信に用いられる．同じ大きさで線数の多いRJ45に置き換わりつつある．

RJE
Remote job entry.コンピュータに実行させるジョブを投入する設備．カードリーダは初期のRJEステーションである．通常はRJEステーションをエミュレートするソフトウェアを意味する．

rlogin
Remote log-in. 遠隔ノードにログオンするためのBerkeley TCP/IPのコマンド．

RLP
Resource Location Protocol. RFC 887で定義されている選択的Internetプロトコル.

RM
Record marking.RPCプロトコルで用いられている技術で，ストリーミングTCPプロトコルにおいてレコードを他のレコードと区別する.

RMS
Record management services.VMSの一般的なI/Oインターフェイスで，QIOコールでローカルデータを，DAPプロトコルでリモートデータをアクセスするのに用いられる.

Routing Table Maintenance Protocol
ルーティングテーブル管理用の AppleTalk プロトコル.

RPC
遠隔手続き呼出し(remote procedure call)を参照.

RPC Tool
Netwise から販売されている RPC機構で，RPC コンパイラが含まれている.

RPCコンパイラ(RPC Compiler)
Netwise RPC Tool の一部として提供されているソースコードジェネレータ. [Netwise RPC Tool]

RPCサーバ制御ライブラリ(RPC Server Control Library)
Netwise RPC Tool の一部で，サーバのアプリケーションプロセスのRPC の部分を管理するルーチン群からなっている.

RPC仕様(RPC Specification)
遠隔手続き呼出しの使用法の観点から，クライアントとサーバのアプリケーションプロセス間のやりとりを記述したもの. この仕様が Netwise RPC Compiler の入力となる. [Netwise RPC Tool]

RPCバインディング(RPC binding)
単一の RPC インターフェイスで遠隔手続きを発行するための，クライアントとサーバのアプリケーション間の論理的な結び付き. [Netwise RPC Tool]

RPCマクロ(RPC macro)
いくつかの特殊なマクロのうちの1つで，Netwise RPC Compiler で作られたソースコードをプログラマがカスタマイズするのに使用することができる. [Netwise RPC Tool]

RR
Receive ready あるいは Resource Record. Receive ready は LLC フィールドの1つで，それを送ったノードがデータを受け取る準備ができていることを示す. Resource Record はドメイン名前システムに記録されている情報. いろいろな種類の resource record が使用可能で，システムは拡張可能である.

RRQ
Read Request. 自明ファイル転送プロトコルのパケットの種類.

RS-232-C
物理インターフェイスの標準で，端末のような非同期装置を接続するのによく使われる. Electronic Industries Associtaion で開発され，DTE と DCE 間の電気的・機械的接続法を規定している.

RSA
Riverst，Shamir および Adleman. 公開鍵暗号方式の特許の開発者. Internetのプライバシー強化メールや X.509 ディレクトリセキュリティの基礎となっている.

RSADSI
RSA Data Security International. RSA によって設立された会社で，彼らの研究の商用化をしている.

RST
Reset. TCP ヘッダのフラグの1つ.

RSTS
Resource-sharing timesharing system. PDP用のオペレーティングシステム.

RSX
もう1つのPDP用のオペレーティングシステム. 互換モードで走っているVMSシステムは RSX実行イメージを実行できる.

RTL
ランタイムライブラリ(Run-Time Library).

RTMP
Routing Table Maintenance Protocol を参照.

RT/PC
RISC アーキテクチャに基づくIBMの32ビットワークステーション.

RTS
Reliable Transfer Service.

RTT
Round-trip transmission. ネットワーク上の遅延の現在値の尺度.

RU

Request Unit．IBMのSNAアーキテクチャの一部．1つのセッションの一方から他方へ一連のRequest Unitが送られる．その後それらがプロトコル階層の上位層で処理される．

RUIP

Remote user information program．RFC 1194で規定されているfingerプロトコルを使ってアクセスされるプログラム．

SAA

Systems Application Architecture．複数のハードウェアプラットフォームやオペレーティングシステム間で共通なユーザ，通信，プログラミングインターフェイスを提供するIBMアーキテクチャ．

SAMBE

Set Asynchrnous Balanced Mode Extended．LLC動作で使われるフレームの種類．

SACK

選択的確認応答(selective acknowledgement)を参照．

SAP

サービスアクセスポイント（Service Access Point)を参照．

SAS

Single Attachment Stationを参照．

SASE

Specific application service element．OSIネットワークアーキテクチャのアプリケーション層の概念．ジョブ転送および操作(job transfer and manipulation: JTM)のような特殊目的のサービスを指す．

SCA

System Communication Architecture．VAXクラスタ用のDECアーキテクチャ．

SCALE

RFC 1072で規定されているTCPウィンドウの倍率オプション．ウィンドウ情報を2の1乗から16乗まで倍率可能で，実効ウィンドウの大きさを大きくすることができる．

SCCP

Signalling connection control part．Q.711を参照．

SCP

サーバ制御手続き(server control procedure)またはセション制御プロトコル(Session Control Protocol)を参照．

SCS

System communication services．VAXクラスタで使われるソフトウェアサービスで，ノード間の通信を提供する．SCSはSystem Communication Architectureの一番下のレベルである．

SCSI

Small computer standard interface．「スカジー」と発音する．ディスクドライブをディスクコントローラにつなぐ標準規格で*，典型的には小規模なマルチユーザコンピュータで使われる．
(*訳注：正確にはディスクコントローラなどをコンピュータにつなぐための周辺バス．)

SDLC

同期データリンク制御(Synchronous data link control)．SNAネットワークで使われるIBMのデータリンクプロトコル．

SDU

サービスデータ単位(Service Date Unit)を参照．

Sequenced Packet Protocol

トランスポート層でのデータの信頼のおける転送のためのXNSプロトコル．

SER

サテライト設備室(Satelite Equipment Room)を参照．

Service Advertisement Protocol

NetWareプロトコルで，サービスの現在のネットワークアドレスを知らせる．

SESQUINET

Sesquicentennial Network．テキサス地域の地域ネットワークで，150周年記念(sesquicentennial celebration)に因んで名づけられた．

set文(set statement)

インターフェイス仕様で使われるC言語の拡張で，インターフェイス変数に値を割り当てるのに用いられる．[Netwise RPC Tool]

SFD

X.400用のメッセージ本体の型．

SFH

Simple Forwarding Header．商業メール中継(Commercial Mail Relay; CMR)で使われる概念で，商業サービスのユーザが，ゲートウェイシステムが転送するのに使う簡単なヘッダ(例えば，宛先ユーザの名前からなる最初のメッセージ行)を付けて専用のメールボックスに送る方式．詳細はRFC 1168を参照．

SG

Study Group.

SGI
Silicon Graphics,Incorporated.

SGML
Standard Generalized Markup Language を参照.

SGMP
Simple Gateway Monitoring Protocol. SNMP の前身.

SIA
Stable Implementation Agreements. OSI Implementors Workshop の製品.

SIG
Special Interest Group.

SIGCOMM
ACM Special Interest Group on Data Communication. データ通信やネットワークに興味をもつ専門家の分科会.

Single Attachment Station
FDDI 用語で, コンセントレータ経由でリングにつながれるステーション.

SLIP
Serial Line IP. シリアル回線上で IP を動かす簡単なプロトコルで, RFC 1055 で規定されている.

small movable cabling system.
IBM のケーブルシステム. 同じ会社で売っている大きくて移動不可能なシステムと対比される.

SMDS
Switched Multi-megabit Data Service. Bellcore によって開発された, データグラムベースの公共データネットワークサービスで, データネットワークの標準として広く電話会社に用いられることが期待されている.

SMI
管理情報の構造 (Structure of Management Information). RFC 1155で規定されている推奨 Internet プロトコル.

S-mode
OSI仮想端末プロトコルで使用される, 2方向交互対話(a two-way alternate dialogue)から構成される端末操作のモデル.

SMP
ステーション管理 (station management)を参照.

SMTP
単純メール転送プロトコル(Simple Mail Transfer Protocol). 電子メールのメッセージを, あるコンピュータから別のコンピュータへ転送するためのInternet標準プロトコル. SMTPでは, 2つのメールシステムがどのように通信し合うかや, メールを転送するために交換する制御メッセージの形式が定められている.

SNA
System network Architectureを参照.

SNADS
SNA distribution service. SNA環境でメッセージを転送するために用いられるアーキテクチャで, X.400に類似している.

snail mail
伝統的な郵便サービス.

SNAP SAP
Subnetwork Access SAP. Service Access Point の特殊な形式で, 論理リンク制御データの情報フィールドの最初の5バイトがプロトコル識別子として使われている.

Sniffer
Network General社作成のネットワークアナライザ. Snifferはこの本でネットワークパケットのスクリーンダンプを作成するのに使用されている.

Sniffer Network Analyzer
Network General社の製品で, 様々な上位／下位層ネットワークプロトコルを監視するのに使われる.

SNMP
単純ネットワーク管理プロトコル (Simple Network Management Protocol). TCP/IPネットワークで使われている標準的なネットワーク管理プロトコル. RFC1098で定義されている.

SNPA-address
SubNetwork Point of Attachment address. ネットワークエンティティに対して利用可能なサブネットワークサービスのアドレス.

SOA
Start of Authority. DNS ゾーンファイルの中の印で, 新しい権限の始まりを示す.

Softswitch
異なるメッセージ処理サービスの間のゲートウェイを作っている会社.

SONET
同期光ネットワーク(Synchronous Optical Network). 光ファイバ(fiber optic)に基づいた回線.

SOSIP
Swedish OSI Profile.

SP
空白(space).

SPARC
Scalable Processor Architecture. Sunによって開発された縮小命令セット(reduced instruction set; RISC) プロセッサであり，AT&TやTexas Instrumentsなどの複数の企業にライセンスされている．

SPF
Shortest Path First. OSPFを参照.

SPP
Sequential Packet Protocolを見よ.

SQL
Structured Query Languageを見よ.

SRC
Source address または Source From Address Correct. Source addressは発信元のデータリンクアドレスである．Source From Address Correct は HYPERchannelのヘッダ内のビットで，from フィールドの32ビットアドレスが，実際にそのメッセージの発信地であることを示している．

SS7
Signalling System 7. ISDNに関連したプロトコル．ISDNネットワークの内側がどのように管理されているかを示している．

SSAP
発信元サービスアクセスポイント (Source service access point). サービスの利用者のアドレス．DSAPも見よ．

SSCP
Ssytem service control point. IBMのSNAアーキテクチャでネットワーク中でアドレス可能な装置．メインフレームの中に存在し，SNAネットワークのそのドメインの中心となる．

SSS
Server session socket. AppleTalk Session Protocolのフィールドで，セッションレベルのパケットが送られたソケットの番号が格納されている．

Standard Control Interface
Netwise RPC Server Control Libraryへの簡単で高レベルなプログラムインターフェイス．[Netwise RPC Tool]

Standard Generalized Markup Language
更新可能な形式のテキストの，表現に関するISO規格．

STATSRV
Statistic Server. RFC 996で定義されているネットワーク統計プロトコル．

STE
Signalling Terminal Exchange. X.25ネットワークでネットワークの境界に置かれる装置．異なるX.25管理ドメイン間の通信は，X.75プロトコルを使ってSTE間で行われる．

STN
Switched Telephone Network.

stored upstream address
トークンリングの概念．トークンリング上のそれぞれのノードは，データを受け取る隣接ノードのアドレスを保持する．

STREAMS
Unixオペレーティングシステムのために AT&Tで開発された機構．STERAMSは一連のソフトウェアモジュールを接続する1つの方法で，互いにメッセージを送り合うことができる．

Structured Query Language
関係データベースシステムと通信するための国際的標準言語．

Sun Microsystems
ワークステーションや Network File Systemの製造会社．

SURANET
Southeastern Universities Research Assosiation Network. NFSnetの地域ネットワークの1つ．[RFC 1177]

SVC
Switched Virtual Circuitを見よ．

SVID
System V interface definition. AT&Tが提案している定義で，System Vの異なる実装の互換性を決定するのに用いられる．

switched virtual circuit
ダイアルアップ電話回線やX.25の呼出しの場合のように，要求に応じて設定される仮想的な回線．permanent virtual circuitを見よ．

SWS
馬鹿げたウィンドウ症候群(Silly Window Syndrome). 利用できるウィンドウがゼロにまで減少してしまうような，TCP/IPで見受けられる現象．RFC 813において，Dr. David Clarkによって説明されている．

SYN
Synchronize. TCPのビットで，コネクションのウィンドウを両端で再同期させる．コネクションを初期化したりリセットするのに用いられ

る.

system

構成要素のインスタンス．ときどきホストや装置と呼ばれる．[SunNet Manager]

System network Architecture

IBMのネットワーキングアーキテクチャ．

Systems network Architecture

IBMネットワークアーキテクチャ．

T1

DS-1 形式のディジタル信号を 1.544 Mbpsで送るディジタル伝送設備．

T3

DS-3形式のディジタル信号を44.746 Mbpsで送るディジタル伝送設備．[RFC 1177]

T.4

Group 3のファクシミリ伝送のCCITT規格．

T.6

Group 4のファクシミリ伝送のCCITT規格．

TAC

Telephone Access Code. Milnetの用語．

target system

被管理オブジェクトが存在するホスト(またはオブジェクト)．[SunNet Manager]

TC

transport connectionを見よ．技術委員会(technical committee)の意味でも使用される．

TCC

Telephone Country Code. 指し示される国を表すX.121アドレスの一部．

TCP

伝送制御プロトコル (Transmission Control Protocol)を見よ．

TCP/IP

伝送制御プロトコル / Internet プロトコル (Trasmission Control Protocol/Internet Protocol). IP上で動作するアプリケーションやトランスポートプロトコルをまとめて短く表すためによく用いられる言葉．これには，FTP, Telnet, SMTP, UDP(トランスポート層プロトコル)などが含まれる．[RFC1177]

TCP-LDP

TCP Extensions for Long Delay Paths. RFC 1072で定義された限定使用Internetプロトコル．

TDF

Time Differential Factor. その地点の時差帯と世界標準時(UTC)との差．

Telenet

US Sprintによって提供されているパケット交換ネットワーク．

Telex

faxや電子メール以前のメッセージ転送機構．

Telnet

遠隔端末接続のためのInternet標準プロトコル．Telnetにより，あるサイトにいるユーザが他のサイトにあるタイムシェアリングシステムに対して，あたかもそのユーザの端末が直接接続されているかのように遠隔コンピュータにアクセスすることができる．

TENEX

ニックネーム(Nickname)やwhoisプログラムの初期バージョンの1つ．

terminate and stay resident

アプリケーションプログラムより前にロードされるDOSプログラム．TSRプログラムは特定のキー入力により起動される．BorlandのSidekickはTSRの例である．

THT

トークン保持タイマ(Token holding timer)．トークンリングおよびFDDIにおける用語で，あるノードがトークンをリングに返す前にデータを送信できる時間の長さを表す．

TID

トサンザクションID(Transaction ID)．

TLI

トランスポートレベルインターフェイス(Transport Level Interface)を参照．

TP

撚り対線(Twisted Pair)．

TPDU

トランスポートプロトコルデータ単位(Transport Protocol Data Unit)．OSIネットワーク中の2つのトランスポート層エンティティの間で交換される情報パケット．PDU参照．

TR

技術報告(Technical Report)．

TransLAN

Vitalink社製の広域拡張イーサネットブリッジ．

traps文(traps statement)

RPCコンパイラに対して，クライアントやサーバスタブ，あるいはディスパッチャのために生成するソースコード中にカスタム化されたエラー処理命令を付け加えるように指示するよう

な，C言語の拡張．[Netwise RPC Tool]

TRT

トークン周回タイマ(Token rotation timer)．トークンリングおよびFDDIにおける用語で，トークンがリングを周囲するのに要する時間の長さを表す．TTRTも参照せよ．

TSR

Terminate and Stay Residentを参照．

TTL

余命(Time to Live)．Internetプロトコルヘッダ中のフィールドで，そのパケットが廃棄されるまでどれだけ生き延びられるかを表す．

TTRT

トークン周回目標時間(Target token rotation time)．FDDIで性能パラメータを指定するのに用いられる用語．TTRTは遅延の期待値の尺度であり，タイムアウトパラメータの設定などに用いられる．

TTX

テレテックス(Teletex)．

TTY

テレタイプ(Teletype)．行指向の端末のこと．

TUP

Telephone User Part．Q.721を参照．

TW

細径ケーブル(Thin Wire)．

TWG

ウロンゴングループ(The Wollongong Group)．ウロンゴンはオーストラリアNew South Wales州東部の都市．

Tymnet

McDonnell Douglas Network Systems Companyにより運用されている公衆パケット交換網．[RFC 1177]

Tymnet

McDonnell-Douglas所有のX.25に基づく公衆パケット交換網．

UA

番号なし確認応答(Unnumbered acknowledgement)．LLC動作において用いられるフレームの種類．

UAM

ユーザ認証方式(User Authentication Method)を参照．

UAPDU

ユーザエージェントプロトコルデータ単位(User Agent Protocol Data Unit)．

UCL

University College London．X.500ディレクトリおよびX.400メッセージ処理システムの分野において活発な研究を行っている英国の研究センター．

UDP

ユーザデータグラムプロトコル(User Datagram Protocol)を参照．

UI

番号なし情報(Unnumbered Information)．LLC動作で用いられるフレームの種類．

UIC

ユーザ識別コード(User identification code)．システム上のすべてのユーザを一意に識別するために用いられる数．

Ultrix

DEC社がVAXコンピュータ用に販売しているUnixのバージョン．

Unibus

11/780および8600 VAXプロセッサで用いられている周辺バス．

Unix

アメリカ電話電信会社(American Telephone and Telegraph)が開発し，商標をもっているオペレーティングシステム．UnixはMulticsの語呂合せである．

USASCII

ASCII文字セットのうち十進128より下に対応する部分．

USC

南カリフォルニア大学(University of Southern California)．

Usenet

Unixユーザのネットワーク．電子メールや掲示板の形で情報を交換することに同意した，ゆるく結合されたノードによる，どちらかというと非公式なネットワーク．

UT

世界標準時間(Universal Time)．

UTC

協定世界標準時間(Coordinated Universal Time)．時刻に関する協定世界標準．現地時間はUTCに時差を加えたものである．

UTnet

テキサス大学ネットワーク(University of Texas Network)*．

(*訳注：東京大学情報ネットワークシステム

(University of Tokyo Network)もUTnetと呼ばれている．)

UTS
　更新タイムスタンプ(Update timestamp)を参照.

UUCP
　Unix間コピープログラム(Unix-to-Unix copy program)．2つのUnixノードの間で情報を交換するのに用いられる標準ユーティリティ．Usenetの基盤として用いられている.

V.21
　一般の交換電話網用の300-bps全二重モデムに関するCCITT規格.

V.22
　一般の交換電話網用の1200-bps全二重モデムに関するCCITT規格.

V.22bis
　一般の交換電話網用の2400-bps全二重モデムに関するCCITT規格.

V.23
　600-および1200-bpsモデムに関するCCITT規格.

V.24
　DTEとDCEの間の回路の定義に関するCCITT規格.

V.27
　専用回線用の4800-bpsモデムに関するCCITT規格.

V.27bis
　専用電話回線用の4800-および2400-bpsモデムに関するCCITT規格.

V.27ter
　一般交換電話回線用の4800-および2400-bpsモデムに関するCCITT規格.

V.29
　4線式専用回線用の9600-bpsモデムに関するCCITT規格.

V.32
　一般および専用電話回線用の9600-bpsまで動作可能な一連の2線式モデムに関するCCITT規格.

V.33
　専用回線用の14.4-kbpsモデムに関するCCITT規格.

V.35
　高速データ伝送に関するCCITT物理インターフェイス規格.

VAN
　付加価値網(Value-Added Network)を参照.

VAR
　付加価値値再販売者(Value-added reseller)を参照.

VAX
　Virtual address extension．DEC社が製造しているハードウェアのシリーズ.

VAX Clusters
　イーサネットではなくCIバス上で動作するDECクラスタ.

VBN
　仮想ブロック番号(Virtual Block Number)を参照.

VINES
　Banyan社が作成したネットワークアーキテクチャ.

VM
　仮想マシン(virtual machine)または仮想記憶(virtual memory)を参照．VMはIBMのオペレーティングシステムの1つで，MVSのようなゲストオペレーティングシステムがその上で動作することができるようにするものである．通常CMSユーザインターフェイスと組み合わせて用いられる.

VMS
　Virtual Memory System．VAXコンピュータ用のDEC固有のオペレーティングシステム.

VMTP
　Versatile Message Transaction Protocol．RFC 1045で定義されているトランスポート層プロトコル.

VSAM
　仮想記憶アクセス方式(Virtual sequential access method*)．IBM環境で直接アクセスファイルに用いられるファイル構成の方式．ISAM(indexed sequential access method：索引順アクセス方式)に似ている．(*訳注：virtual storage access methodが正しい．)

VT
　仮想端末(Virtual Terminal)を参照.

VT100
　DECの端末のシリーズ.

VTAM
　Virtual Telecomminication Access Method．SNAネットワークに対するインターフェイスを提供しているIBMのソフトウェアシステム.

WAN
広域ネットワーク(Wide-area network). 職場ネットワーク(work area network)すなわち職場における小規模なサブネットワークを指すのに用いられることもある.

WESTNET
全米科学財団が資金を出している地域TCP/IPネットワークの1つで，アリゾナ州，コロラド州，ニューメキシコ州，ユタ州，ワイオミング州をカバーしている. [RFC 1177]

WG
作業部会(Working Group).

whois
人やドメイン，ネットワーク，ホストなどのInternetエンティティに関してNICが保持しているデータベースに問合せを行うためのInternetプログラム. 人に関する情報には会社名，住所，電話番号，電子メールアドレスがある. [RFC 1177]

WIA
Working Implementation Agreement. Stable Implementation Agreement(SIA)の状態に到達しなかった，一時的な合意.

WKS
周知のサービス(well-known service). TCPまたはUDP上のサービスで，周知のポート番号を用いており，したがって，アプリケーションがある時点でどのポート番号を用いているかあらかじめ知っていなくてもアクセスすることができるもの.

WORM
Write once/read many. CD-ROMのディスクとは異なり，自分で書き込むことのできる光学ディスクの種類. また，釣りでも用いられる*.
(*訳注：餌に用いる虫のこと)

WRL
Western Research Laboratoryを参照.

WRQ
書込み要求(Write Request). Trivial File Transfer Protocol(RFC 783)のパケットの種類.

WYSIWYG
What You See is What You Get.

X.3
パケット組立／分解装置(PAD)に関するCCITT規格.

X.12
電子データ交換(Electronic Data Interchange)に関するANSIの委員会.

X.21
回線交換ネットワークに関するCCITT規格.

X.21bis
公衆データネットワーク上での同期式V-シリーズモデムの使用.

X.25
パケットモードで動作し，専用線により公衆データネットワークに接続されている端末におけるDTEとDCEの間のインターフェイスに関するCCITT規格.

X.28
非同期式端末がX.3 PADと通信するためのCCITTプロトコル.

X.29
同期式DTE(ホスト)がX.3 PADを制御し，通信するためのCCITTプロトコル.

X.75
独立したX.25ネットワーク間の相互接続に関するCCITT規格.

X.81
ISDN回線交換網と回線交換公衆網の間のインターネットワーキング.

X.110
公衆データネットワーク上のルーティング方針に関するCCITT規格.

X.121
公衆データネットワークにおけるCCITTの番号計画.

X.200
OSI参照モデルのCCITT版.

X.208
OSI ASN.1のCCITT版.

X.209
OSI ASN.1基本符号化規則(BER)のCCITT版.

X.211
CCITTアプリケーション用のOSI物理サービスの定義.

X.212
CCITTアプリケーション用のOSIデータリンクサービスの定義.

X.213
CCITTアプリケーション用のOSIネットワーク層サービスの定義.

X.214
CCITTアプリケーション用のOSIトランスポートサービスの定義.

X.215
　CCITTアプリケーション用のOSIセッションサービスの定義.
X.216
　CCITTアプリケーション用のOSIプレゼンテーションサービスの定義.
X.217
　CCITTアプリケーション用のOSI ACSEの定義.
X.218
　ISO 9066-1(テキスト通信——信頼できる転送)に対するCCITTの相当品.
X.219
　ISO遠隔操作サービス要素(Remote Operations Service Element: ROSE)に対するCCITTの相当品.
X.220
　CCITTアプリケーションにおけるX.200-シリーズプロトコルの使用に関するCCITT仕様.
X.223
　OSIコネクションモードサービスを提供するためにX.25を使用すること.
X.400
　メッセージ処理に関するCCITT規格.
X.402
　CCITTメッセージ処理システム：全体的なアーキテクチャ.
X.403
　CCITTメッセージ処理システム：適合性試験.
X.407
　CCITTメッセージ処理システム：抽象的サービス定義に関する取決め.
X.408
　CCITTメッセージ処理システム：符号化された情報種別の変換規則.
X.411
　CCITTメッセージ処理システム：メッセージ転送システム：抽象的サービス定義と手続き.
X.413
　CCITTメッセージ処理システム：メッセージ格納，抽象的サービス定義.
X.419
　CCITTメッセージ処理システム：プロトコル仕様.
X.420
　CCITTメッセージ処理システム：個人間メッセージシステム.
X.500
　ディレクトリ情報に関するCCITT規格.
X.501
　CCITTディレクトリ：モデル.
X.509
　CCITTディレクトリ：認証の枠組.
X.511
　CCITTディレクトリ：抽象的サービス定義.
X.519
　CCITTディレクトリ：プロトコル仕様.
X.520
　CCITTディレクトリ：主な属性の種類.
X.521
　CCITTディレクトリ：主なオブジェクトクラス.
XDR
　外部データ表現(External Data Representation)を参照.
Xerox
　いろいろなコンピュータや画像製品の製造業社．さらに広い範囲のいろいろな製品の発明者.
Xeroxネットワークシステム(Xerox Network System)
　上位(第3層および第4層)プロトコルにいくつかのアプリケーションを加えたもので，通常イーサネットと組み合わせて用いられる．DECnetやTCP/IPに代わるもの．第3層および第4層のプロトコルはNovellのNetWareの基礎となっている.
Xeroxパロアルト研究センター (Xerox Palo Alto Research Center)
　PostScript，AppleのMacintosh，イーサネットなど，現在の多くの技術の開発を先導してきたXerox社の研究所.
XID
　Exchange identification．新しいノードが物理媒体に接続されたときに用いられるHDLCフレーム．XIDフレームには，ノードIDやコネクションの検証パスワードなどの情報が格納されている.
XMI
　VAX6200シリーズの並列プロセッサにおいて，CPU，BIバスおよびメモリを接続するのに用いられている100Mbpsのバス.
XNS
　Xeroxネットワークシステム(Xerox Network

System)を参照.

XNSメール転送プロトコル(XNS Mail Transfer Protocol)

XNSにおけるメッセージ処理サービス.

アイコン(icon)

閉じられているウィンドウを表すためにワークステーションで用いられている小さな絵のオブジェクト．ユーザがアイコンを指し，マウスボタンをクリックするとウィンドウが開く.

アウトゴーイングコネクションタイマ (Outgoing connection timer)

セション制御用語．セション層がトランスポート層に対して接続要求を発したとき，このタイマを起動する．このタイマが終了するまでに，受諾または拒否が得られないと，セション制御は切断を発してエンドユーザに知らせる.

アクセス制御セット(Access Control Set)

オブジェクトの識別子およびあるエンティティ(例えばノードやファイル)に対するそれらの権限の集合.

アクセス法(access method)

ファイル中の情報にアクセスする方法．B-Treeはアクセス法の一例である.

アクティビティデーモン(activity daemon)

このデーモンが動作しているホストから送られてきたアクティブエージェント要求のリストを保持しているプロセス．このデーモンは定期的にエージェントに問合せを発行して，それらがまだその要求をサービスしていることを確認する．[SunNet Manager]

アクティブコンバータ(active converter)

ある通信信号インターフェイスを他のものに変換するために用いられる装置.

アクティブデバイス(active device)

それ自身が電源をもつデバイス.

アクティブモニタ存在(Active Monitor Present)

トークンリング上のアクティブモニタが3秒ごとに出すパケット.

アドレス(address)

インターネットネットワーキングではこの単語は2通りの使い方がある.「電子メールアドレス」と「インターネットアドレス」である．電子メールアドレスは，特定の個人へのメッセージを送るために電子メールプログラムに与える文字列である．[RFC1177] インターネットアドレスも参照.

アドレス解決プロトコル (Address Resolution Protocol)

IPアドレスをMACアドレス(イーサネットアドレスやその他のサブネットワークアドレスなど)に翻訳するTCP/IPプロトコル.

アドレス空間 (address space)

インターネットワークのような一元的集合を形成するアドレスの集合.

アドレス割当て機構 (addressing authority)

ドメイン内でアドレス割当ての責任をもつ団体.

アプリケーション (Application)

ユーザのためにある機能を実現するプログラム．注文受付システム[order entry system]やワードプロセッサはアプリケーションの例である.

アプリケーション環境仕様 (Application Environment Specification)

Open Software Foundationの目標の1つ.

アプリケーション制御サービス (Application Control Service)

NASの概念である．NASの環境でアプリケーション間の相互作用に用いられる規則である.

アプリケーション層

ネットワークプロトコル階層の一番上の層である．アプリケーション層は仕事のセマンティクスに関与している．例えば，ある外部のノードから，キーの値を用いてファイルから特定の記録を得ることは，アプリケーション層が関与することである．そのデータをどのように表すか，どのようにその外部ノードに到達するかは，ネットワークのより低い層の課題である.

アプリケーションターミナル

スレーブノード上でアプリケーションにアクセスされるLATオブジェクト．例えば，ターミナルサーバに接続されているプリンタは，プリントスプーラに対するアプリケーションターミナルである.

アプリケーションプログラミングインターフェイス (Application Programming Interface)

2つのプログラムの間の呼出し構造(calling structure)の仕様．通常，一般的なアプリケーションプログラムと，通信サポートのような特定の支援サービスとの間のものである.

アプリケーションプロセス

1つの計算機の上で実行される分散アプリケーションの構成要素．プロセスと同じ意味であ

る．[Netwise RPC Protocol]

アンパック(Unpack)
遠隔手続き呼出しで用いられる用語で，データを機種独立の形式から，特定のコンピュータで用いられる形式に翻訳することを指す．

アンパック手続き(unpack procedure)
Netwise RPCコンパイラにより生成される手続きで，PDUを標準形式からホストプロセッサ用のCのデータ型に復号化する．[Netwise RPC Tool]

暗黙の結合(implicit binding)
結合パラメータ宣言を行うのではなく，結合外部変数宣言を用いることで結合宣言を行う方式．[Netwise RPC Tool]

イエローページ(Yellow Pages)
ネットワークファイルシステム(Network File System)においてマスタから受信者に情報を伝搬させる一連のサービス．複雑なネットワークにおいてシステムファイルを管理するのに用いられる．イエローページはセールス用語としてはネットワーク情報サービス(Network Information Service)として知られている．

イーサネット(Ethernet)
Intel，XeroxおよびDECが共同で開発し，後に802.3標準としてIEEEにより採用されたデータリンクプロトコル．DECnet，TCP/IPおよびXNSなどの上位層プロトコルが，イーサネットを下位の伝達機構として用いる．トークンリング，DDCMPやSDLCのような，他のデータリンクプロトコルと対比される．

イーサネットコントローラ(Ethernet contoller)
コンピュータがイーサネットのサービスにアクセスできるようにするデバイスコントローラ．通常CSMA/CDプロトコルはコントローラに実装されるので，CPUがプロトコルの詳細に関与する必要はない．

イーサネットバージョン2.0(Ethernet Version 2.0)
イーサネットのもともとの仕様の第2版であり，IEEE 802.3標準とは少し違う．

イベント(event)
オブジェクト属性の状態にある種の変化が起こること．

イベントシンク(event sink)
重要なイベントが送られるべきネットワーク上の宛先．シンクの例として，ファイル，プログラム，端末などが挙げられる．

イベントディスパッチャ(event dispatcher)
イベント報告の待合せの役割を果たすプロセス．管理アプリケーションはこのデーモンに対して，ある基準に基づいて選び出した報告のコピーを送ってもらうことができる．[SunNet Manager]

イベント報告(event report)
被管理オブジェクト上でイベントが検出されたときに，エージェントが送る通知．[SunNet Manager]

イベントログ(event log)
ディレクタに保存されるべき重要なイベントの記録．

インスタンス(instance)
特定のオブジェクトを表すデータ．MDBが用いるとき，インスタンスの定義はインスタンスレコードと呼ばれる．[SunNet Manager]

インスタンスファイル(instance file)
エージェント要求と要素インスタンスレコードを集めたもの．[SunNet Manager]

インターネット(internet)
ゲートウェイやルータで互いに接続されたネットワークの集まり．ゲートウェイを用いて電子メールでアクセスすることのできる不均質なネットワークの混合体をしばしば表す．

インターネットアドレス(internet address)
インターネット中のホストを識別するために割り当てられた番号．ネットワーク番号，サブネット番号(無いこともある)，ホスト番号という，2つまたは3つの部分からなっている．[RFC 1177]

インターネットワーク (internetwork)
データリンクとデータリンク間の経路づけを行うネットワーク層プログラムの集まり．

インターネットワークアドレス (internetwork address)
ネットワーク番号と，そのネットワーク上での局所的なアドレスによって構成されるアドレス．ネットワーク層でパケットを最終的な目的値に送るための経路づけに用いられる．

インターネットワークデータグラムプロトコル(Internetwork Datagram Protocol)
XNSのネットワーク層プロトコル．

インターフェイス仕様(interface specification)
1つ以上のインターフェイス変数にset文を用いて値を割り当てること．[Netwise RPC Tool]

インターフェイス変数(interface variable)
Netwise RPC Compilerの動作を制御する変数．インターフェイス仕様においてset文を用いて定義される．[Netwise RPC Tool]

インデックス(index)
データへの直接アクセス方式の1つ．インデックスは，キーの値とそのキー値をもつデータを格納している表中の列へのポインタをもっている．また，インデックスには，実際の表の記憶構造の一部である1次インデックスと，元の表へのポインタをもつデータベース中の別の表である2次インデックスとがある．

インデックスマクロ(index macro)
配列の中への添字を指定する表現マクロ．インデックスマクロはシャープ記号(#)で始まる．[Netwise RPC Tool]

隠蔽された(opaque)
プログラムによって解釈されないデータ型．セキュリティアプリケーションの信任状やNFSのファイルハンドルを格納するのにしばしば用いられる．

ウィンドウ(window)
あるTCPコネクションにおいて確認応答を待たずに送ることのできるデータの量．

永続的オープン方式(persistent open style)
バインディングを制御する方法の1つ．遠隔手続き宣言においてこの方式が使われると，Netwise RPCコンパイラは永続的バインディングを確立するコードを作る．[Netwise RPC Tool]

永続的クローズ方式(persistent close style)
バインディングを制御する方法の1つ．遠隔手続き宣言で使われると，Netwise RPCコンパイラは既存の永続的バインディングを使い，遠隔手続きに戻るときにそれを終了するようなコードを作成する．[Netwise RPC Tool]

永続的バインディング(persistent binding)
1つ以上の遠隔手続き呼出しで使うように，ネットワーク上のアプリケーションプロセス間で確立されているバインディング．

永続的利用方式(persistent use style)
バインディングを制御する方法の1つ．遠隔手続き宣言においてこの方式が使われると，Netwise RPCコンパイラは既存の永続的バインディングを使うコードを作成する．[Netwise RPC Tool]

エージェント(agent)
管理機能に応答するエンティティの部分を指すネットワーク管理用語．

エージェントサービス(Agent Services)
エージェントから見たときに，エージェントとマネージャが通信するのを可能にしている関数ライブラリ．[SunNet Manager]

エージェントスキーマ(agent schema)
グループと，エージェントから利用可能な属性を記述したもので，エージェントの誤りの可能性を列挙したリストを含む．この記述によって，マネージャプロセスは特定のグループや属性に対して要求を出すことができる．[SunNet Manager]

遠隔手続き(remote procedure)
サーバアプリケーションプロセス中にユーザが記述した手続きで，クライアントアプリケーションプロセスから呼び出される．[Netwise RPC Tool]

遠隔手続き呼出し(remote procedure call)
ノードが遠隔マシンで実行される手続きを呼び出すための一連のネットワークプロトコル．Netwise RPC Tool，HP/Apollo RPC，SunのNFS RPCなどがこのようなプロトコルの例である．

エンディアン(endian)
コンピュータがマルチオクテットのデータ(例えば4バイトの整数)を格納する方法．ビッグエンディアン(big endian)およびリトルエンディアン(little endian)参照．

エンドシステム(End System)
アプリケーションが動作するOSIシステム．エンドシステムにはOSI 7層すべての機能がある．基本的には，Internetホストと等価である．[RFC 1136]

エンドポイント(endpoint)
結合終端(binding endpoint)参照．

オートボー(autobaud)
受信側にあるモデムが発信モデムの伝送速度を自動的に検出する機能．

オープンシステム間相互接続(Open Systems Interconnection)
不均質なオープンネットワークアーキテクチャに関するISO標準．

オープンシステム間相互接続環境(Open Systems Interconnection Environment)
世界的規模のオープンシステム集合体．基本的にはInternetと同じ．[RFC1136]

オープンソフトウェア基金(Open Software Foundation)
　オープンソフトウェア環境に関する仕様を開発するために，Digital，IBM，その他4つのベンダによって設立された非営利組織．

オープンネットワークコンピューティング(Open Network Computing)
　Network File Systemを含むプロトコルの族に対するSunのマーケティング用語．

階層的ルーチング(hierarchical routing)
　ドメインに基づくルーチング．ドメイン間のルータはデータを正しいドメインに届ける責任だけをもっている．次いで，ドメイン内ルータがドメイン内でのルーチングの責任を果たす．

階層データベース(hierarchical database)
　データを表の形にではなく階層的に構造化するデータベース．プログラマは特定のデータの列を取り出すために階層を指定する．IMSは階層データベースの例である．

外部データ表現(External Data Representation)
　NFSの一部としてSun Microsystemsにより開発されたプレゼンテーション層プロトコル．

外部変数モード(external variable mode)
　遠隔手続きにより外部変数の値がどのように用いられるかを指定するC言語の拡張．in, out, in outのいずれか．[Netwise RPC Tool]

回路(circuit)
　ネットワーキング用語．ネットワーク上の2ユーザ間の論理的データストリームを意味する．1つの物理リンク上に複数の仮想回路が存在し得る．

書込みロック(write lock)
　排他ロックとしても知られる．ロックしたデータを他から読み書きされるのを防ぐ*．
　(*訳注：書込みロックというと，普通はロックされていても読出しはできるものを指す．)

カスタム命令(custom instructions)
　クライアントスタブ，サーバスタブ，ディスパッチャの中に挿入された，ユーザによって書かれた命令．Netwise RPC規格にHooks,Entry,Exit,Trapsステートメントを付け加えることでつくられる．[Netwise RPC Tool]

仮想回線(virtual circuit)
　通常トランスポート層により提供されるサービス．仮想回線のユーザは，データをパケットに入れたり，誤り回復をしたり，データが失われたり，経路判断などをしたりすることなく，相手ユーザに対してデータを送ることができる．

仮想記憶(virtual memory)
　オペレーティングシステムにおける概念で，プログラムのアドレス空間を指すものである．ページングシステムが必要に応じて限られた物理メモリに仮想記憶をマップする．

仮想端末(Virtual Terminal)
　あるシステム上のユーザが他のシステムの会話的セッションにログオンすることができるようにするOSIサービス．

型定義(type definition)
　ユーザ定義の型を宣言する文で，struct，union，enum，typedef，またはmacrodef文を含む．[Netwise RPC Tool]

カーニング(kerning)
　印刷用語で，文字間，単語間の間隔の大きさを調節して均等な配置を実現すること．

カーネル
　オペレーティングシステムで用いられる用語．カーネルは，オペレーティングシステムの低位の機能をユーザシェルのような高位のインターフェイスから隠蔽する．

環状マーク(annular mark)
　イーサネット同軸ケーブル上のマークで，ケーブル上のタップに必要な2.5mの間隔を示す．

関数(function)
　あるデータを入力として受け取り，値を返すもの．例えば，問合せ(query)言語は日付を受け取り，その日付の曜日を返す関数をもっている．

管理領域(administrative domain)
　単一の組織あるいは管理権限によって運営される，エンドシステム，中間システム，サブネットワークの集合．領域を構成する要素は，それらの間では互いに大きな信頼をもって相互運用されるが，異なる管理領域間では互いに懐疑的な方式で相互運用される．

関連制御サービス要素 (Association Control Service Element)
　アプリケーションエンティティがアソシエーションを形成できるようにするOSIアプリケーション層の機能のコアセット[core set]．

規格(standard)
　2つの構成要素の間の標準的なインターフェイス．規格で素敵なことは，たくさんの中から選べることである．

ギガバイト(gigabyte)
　十億バイトのデータ．

キャッシュ(cached)
ディスクに書き込むデータを，メモリ上にとどめておくこと．データをキャッシュしておくことにより，ディスクの読み書きの回数を軽減することができる．

キャラクタレパートリー(character repertoire)
OSI仮想端末プロトコルで用いられる．端末が表示できる文字セット．

キャリア監視・複数アクセス／衝突検出(Carrier Sense-Multiple Access/Collision Detect, CSMA/CD)
複数の計算機が，1つの物理媒体をアクセスしようとする場合に用いられる方法で，イーサネットで使われている．

鏡像化(mirrored)
あるディスクドライブにおいて，データのコピーが別のドライブにある場合，そのドライブは鏡像化されているという．この場合，一方が故障してももう一方が処理を続けることができる．

共通ドメイン (common domain)
上位レベルドメインのメンバでない管理ドメイン．共通ドメインはルーティング階層の最上位レベルである．共通ドメインの上に存在する単一のドメインというものはない．この意味で，ルーティング階層というのは，それぞれの頂点に共通ドメインをもつ事実上複数の階層である．[RFC 1136]

均質(homogeneous)
同じであること．

クッキー(cookie)
サーバからクライアントに送られたファイルハンドルを保持するためにNFSで用いられる隠蔽(opaque)データ．

クライアント(client)
別のモジュールのサービスを利用するモジュール．例えばセッション層はトランスポート層のクライアントである．

クライアントアプリケーションプロセス(client application process)
サーバアプリケーションプロセスで満足されるような遠隔手続き呼出しを発行するアプリケーションプロセス．クライアントプロセスと同義．[Netwise RPC Tool]

クライアントスタブ(client stub)
コンパイラによって生成されたソースコードで，ユーザが書いたクライアントコードがリモートプロシージャをまるで自計算機上のプロシージャのように呼べるようになる．プログラマは1つ以上のクライアントスタブをクライアントプログラムとリンクする．[Netwise RPC Tool]

クライアントソースファイル(client source file)
RPCコンパイラによって作られたCのソースコードファイルで，クライアントスタブコードとパックおよびアンパックプロシージャを含む．[Netwise RPC Tool]

クライアントヘッダファイル(client header file)
Netwise RPC コンパイラによってクライアントのために生成されるテキストファイルで，C言語のマクロ定義を含んでいる．クライアントソースコードはCコンパイラでコンパイルを行う際，クライアントヘッダファイルを取り込む．[Netwise RPC Tool]

クラスタ(cluster)
全体として1つの要素(element)を示すような，MDB中のレコードの集まり．クラスタレコードはインスタンスファイルの中で定義される．クラスタレコードとも，インスタンスレコードとも呼ばれる．[SunNet Manager].

また，ファイルシステムの概念でもある．ディスクは一連のクラスタでできている．クラスタは，必要に応じてファイルやディレクトリに動的に割り当てられる．ブロックと似ている．

グラフィカルカーネルシステム(Graphical Kernel System)
2DオブジェクトFの生成と操作のためのISO標準．

クリアリングハウス(clearinghouse)
DNSで使われているような一連の名前．その時点での資源の位置を知るために，ユーザはクリアリングハウス(ネームサーバとしても知られる)に問い合わせることになる．

グリフ(glyph)
絵による表現．SunNet Managerコンソール(SunNet Manager Console)は3つの形態のグリフを提供する：固定サイズ画像(構成要素(components)，ビュー(views)，現在のビューの中に両端の要素を含まないコネクション)，操作ハンドル(grab handles)をもつ可変長ベクトル(バス)，そして可変長ベクトル(現在のビューの中に両端の要素を含むコネクション)．[SunNet Manager]

グループ
関連する属性の集まり．エージェント(agent)ライタにより定義され，エージェントスキーマにより指定される．[SunNet Manager]

クレジット(credit)
ネットワークプロトコルで使われるフロー制御メカニズム．ノードはクレジットをもっているときにのみパケットを送ることが許される．もしもっていなければ，相手が送ってくるのを待たねばならない．

クローズドユーザグループ(Closed User Group)
CCITT(X.25 とISDN)でのデータ通信における概念．限られたユーザ(ネットワークアドレス)だけが局所コネクション(local connection)にアクセスできる．

グローバルエンティティ(global entity)
管理目的のエンティティ木のトップレベル．ノードはルーティング，MOP，またはLATのようなモジュールを含むサブエンティティをもつグローバルエンティティの例である．

経路情報プロトコル(Routing Information Protocol)
Internet上で用いることのできるプロトコルで，ゲートウェイ間でルーチング情報をやりとりするだけのものである．多くのLANやいくつかのNSFnetの地域ネットワークで用いられている．

結合(bind)
SNAの用語であり，2つの論理的なユニットの間にセッションを確立することを指す．

結合づけ(binding)
遠隔手続き呼出し(remote procedure call)で使われる概念．2つの離れたプログラムはコネクションを起動し，要求コマンドを交換することで互いに結合する．RPC bindingも参照．

結合記述子(binding descriptor)
Netwiseの定義したbinding_descという型をもち，特定の結合終端に関する情報を含んでいる．[Netwise RPC Tool]

結合クラス(binding class)
プロセス結合宣言に現れるパラメータや変数に対する修飾子(qualifier)である．結合クラスにはバインドイン(bind in)，バインドアウト(bind out)，そしてバインドイン/アウト(bind in/out)がある．[Netwise RPC Tool]

結合形式(binding styles)
Netwise RPC コンパイラでサポートされている，結合を実現するための方法．non-persistent, persistent-open, persistent-use, persistent-closeの4つがある．[Netwise RPC Tool]

結合終端(binding endpoint)
結合の一方の端を表す内部表現．[Netwise RPC Tool]

結合種別(binding type)
Netwiseの定義したC言語の型で，結合宣言中のパラメータや外部変数に用いられる．新しい結合に用いられるbinding_infoと，すでにある結合に用いられるbinding_descがある．[Netwise RPC Tool]

結合情報(binding information)
ネットワーク上にあるサーバのアプリケーションプロセスを識別するのに用いられるデータ構造．[Netwise RPC Tool]

結合宣言(binding declaration)
RPC定義中で，結合の情報を記述した文．パラメータまたは外部変数と，その結合種別および結合クラスからなる．[Netwise RPC Tool]

結合パラメータ(binding parameter)
結合宣言の中で指定されるパラメータや外部変数．[Netwise RPC Tool]

ゲートウェイ(gateway)
ゲートウェイには，ともにネットワークで用いられる，2つの少し矛盾する定義がある．一般的な意味では，ゲートウェイは2つの異なるネットワークをつなぐ計算機を指す．通常，これはSNAとDECnetのような2つの異なる種類のネットワークを意味する．しかし，TCP/IP用語では，ゲートウェイは同じネットワークプロトコルを走らせているいないにかかわらず，別々に管理されている2つのサブネットワークを結び付けるものである．

ケーブルパッチパネル(cable patch panel)
サテライト設備室(satellite equipment room)に置かれた装置で，サテライトとオフィスをつなぐ回線やサテライトどうしをつなぐ回線を相互接続するのに使われる．

検証関数(verify function)
要求をディスパッチする前にその要求の正当性を検査するエージェントルーチン．[SunNet Manager]

限定使用プロトコル (Limited Use Protocol)
インターネットの標準に関するプロトコルの状態．これらのプロトコルは，限定された環境でのみ用いられることを想定している．これは，そのプロトコルが実験状態である，特殊な用途

である，機能が限られている，歴史的な経緯，などによるものであろう．

減衰
信号の損失レベルであり，通常dBという単位で表される．

減衰特性
信号がケーブルの上を伝搬していくとだんだん弱くなって(減衰して)いく．媒体の減衰特性とは，その信号が減衰する速さである．

コアゲートウェイ(core gateway)
歴史的には，BBNのInternet Network Operations Centerによって運用されているゲートウェイ(ルータ)のうちの1つのこと．すべてのグループがコアゲートウェイから自分達のネットワークへの経路を広告しなければならないという意味で，コアゲートウェイシステムはInternetルーティングの中枢をなしている．[RFC 1177]

広域ブリッジ(wide-area bridge)
T1回線やダイアルアップ回線のような広域通信回線を用いて動作するMAC層ブリッジ．

広告(advertising)
サービスがその存在をネットワークに知らせることを可能にするプロセス．典型的にはLANベースのマルチキャストにより提供される．

更新タイムスタンプ(Update timestamp)
エントリが最後に更新された時を表すもの．

構成要素(component)
コンピュータ，オペレーティングシステム，ゲートウェイなどのような，ネットワークの要素の一種類．[SunNet Manager]

構造ファイル(structure file)
エージェントと要素の構造の定義の集まり．[SunNet Manager]

高度プログラム間通信(Advanced Program-to-Program Communication)
IBMのシステムネットワークアーキテクチャにおいてピアー間通信に使われるプロトコル．

国際標準化機構(International Organization for Standardization)
国際的な標準作成機関で，開放システム間接続のネットワークアーキテクチャに関して責任をもっている．

国立標準技術研究所(National Institute for Standards and Technology)
標準の制定の援助を行うアメリカ合衆国の政府団体．以前は国家標準局(National Bureau of Standards)といった．

国家標準局(National Bureau of Standards)
国立標準技術研究所(National Institute for Standards and Technology)を見よ．

コネクション(connection)
2つの要素(element)をつなぐポイント間ネットワーク回線．これには，正規型と単純型の2つの型がある．正規型コネクションは，性質(properties)をもったり，エージェントによって管理され得る完全なオブジェクトである．単純型コネクションは，ユーザに便利なようにグラフィカルな表現はもっているが，データベース表現はもたない(単純型は性質をもたないし，エージェントによって管理することもできない)．[SunNet Manager]

コネクションマネージャ(connection manager)
VAX Cluster や LAVC において，状態遷移を管理してクラスタの完全性を保つソフトウェア構成要素を指すクラスタ用語．

コプロセッサ(coprocessors)
コンピュータに付け加えるボードもしくはチップで，メインCPUの行うサービスを補足する．

コールスタブ(called stub)
RPC機構で使われる小さなコードで，RPCサーバ上の手続きから呼出しが遠隔計算機に対するものであることを隠す．

コールバック(callback)
RPC用語で，チェーンされる呼出し(chained call)の呼出し先がもとの要求を出したのと同じアプリケーションプロセスであるような特殊な場合をいう．[Netwise RPC Tool]

コンカレンシ(concurrency)
複数のユーザが同じリソースにアクセスしようとするとき．コンカレントな環境では，ロックマネージャが資源の一貫性管理の問題を処理する．

混合型バッファリング形式 (mixed buffering style)
引数あるいは結果のPDUは1つの連続した主記憶上のバッファに格納され，要求あるいは応答のPDUはパケットのバッファリング形式に従って格納されるというバッファリングの方式．[Netwise RPC Tool]

混雑(congestion)
ある回路に対する過大な通信負荷．

混成ドキュメント(Compound Document)
テキスト，グラフィックス，音声のような複数の種類の情報の入ったドキュメント．

混成ドキュメントアーキテクチャ(Compound Document Architecture)
混成ドキュメント(compound documents)の生成，格納，操作のためのDECのアーキテクチャ．

コンソール(console)
SunNet Managerとともに用いられるウィンドウベースの管理アプリケーション．[SunNet Manager]

コンピュータ支援ソフトウェア工学(Computer-Aided Software Engineering)
ソフトウェア設計のためのソフトウェア．ソースコード管理，データ辞書，エンティティ関係表示ツールなどはすべてその例である．

コンファレシング(conferencing)
参加者が文書(notes)を掲示できるような通信ソフトウェアに対する用語．参加者が明示的に宛先を書かなくてよい点で電子メール[electronic mail in that participant do not have to be explicitly addressed]と対照される．掲示板としても知られる．

最善努力の伝達サービス (Best-effort delivery service)
IPXネットワーク層のように，データを配達しようとするが，回線故障のようなエラーが起こったときには回復しようとしないネットワークモジュール．

サテライト設備室(Satelite Equipment Room)
高速，低速データや音声通信のためのパッチパネルや，ターミナルサーバやリピータ，マルチポートトランシーバのようなアクティブな装置の収められた部屋．

サーバ(server)
ワークステーションにサービスを提供するコンピュータ上のプログラム．ファイル，データベース，印刷，通信などは，サーバの種類のごく一部を挙げたものである．

サーバスタブ(server stub)
Netwise RPCコンパイラで生成されるCのソースコードファイルで，遠隔手続きをあたかも局所的に呼ばれたかのように動作させるものである．プログラマは1つ以上のサーバスタブをサーバプログラムとリンクする．[Netwise RPC Tool]

サーバ制御手続き(server control procedure)
Netwise RPC で使われるプログラム．

サーバソースファイル(server source file)
Netwise RPCコンパイラで生成されるCのソースコードファイルで，サーバスタブコード，ディスパッチャ，パックおよびアンパック手続きが含まれている．[Netwise RPC]

サーバヘッダファイル(server header file)
Netwise RPCコンパイラでサーバ用に生成されるテキストファイルで，C言語のマクロ定義からなり，命令や宣言を含んでいる．[Netwise RPC Tool]

サービスアクセスポイント(Service Access Point)
サービスの利用者のためのアドレス．

サービスクラス(service class)
LAT アーキテクチャでの概念で，対話型端末やウィンドウワークステーションのような特定のアプリケーション群に対する拡張を可能にしている．

サービス総合ディジタル網(Integrated Services Digital Network)
音声とデータを共通の伝送機構に統合することのできるような，最近の国際通信規格．

サービスデータ単位(Service Data Unit)
ある層がクライアントから受け取り，ネットワークに送出する情報．ヘッダや制御情報と対比される．

サブネット(subnet)
すべてのノードが1ホップ先に接続されているような，あらゆるネットワーク技術を指すのに使われる用語．いいかえると，サブネットのユーザは，そのサブネット上の他のすべてのノードと直接通信できる．サブネットはX.25，イーサネット，トークンリング，ISDN，point-to-pointリンクなどがあり得る．一連のサブネットは，ルーティングやネットワーク層とともに，ネットワークを形成する．

サブネットフィールド(subnet field)
インターネットアドレスの中の，サブネット番号を示すビットの領域．このフィールドを構成するビットは必ずしもアドレス上で連続している必要はない．[RFC 950]

サブネットワーク(subnetwork)
ネットワーク層のエンティティ間に直通の経路を提供する通信媒体．これは，point-to-pointリンク，LAN，公衆データネットワークなどによって，実現できる．注意すべき点は，Internetのサブネットとは異なり，OSIでのサブネットワークは必ずしもアドレス階層に反映さ

れている必要はない．したがって，Internetでのサブネットという用語がもつ二重の意味(単一IPホップ，アドレス階層の一部)は，OSIの世界では成り立たない．[RFC 1136]

サブベクタ(subvector)
MACフレームの一部．例えば，リングの初期化を要求するトークンリングコマンドは，アダプタソフトウェアレベル，上流の近傍のアドレス，その他のいくつかのフィールドといったサブベクタを含んでいる．

シェル(shell)
オペレーティングシステムのユーザインターフェイスを指すのに通常用いられる用語．Unixシステムでは，C shell や Bourne shell が基本的なユーザインターフェイスである．コンピュータと低レベルのやりとりをする kernel と対比される．

識別子付きユニオン(descriminated union)
識別子付きユニオンとは型の1つで，識別子と，識別子の値によりあらかじめ設定された型の集合の中から型を1つ選択したものからなる．識別子の型は整数，符号なし整数，または論理値(boolean)のような列挙型のどれかである．構成要素となる型(component types)はユニオンの「腕」と呼ばれ，符号化法を表す識別子が前に付けられる．[RFC 1014]

実験プロトコル(experimental protocol)
インターネット標準の分類．実験プロトコルとは，進行中の研究プロジェクトの一部として運用サービスの提供とは関係なく開発されたプロトコルである．これらのプロトコルは後の段階でサービスプロトコルとして提案されるかも知れないが，あるプロトコルを実験プロトコルと指定することは，ときにはそのプロトコルが成熟しているにもかかわらず，運用上の使用を目的としないことを意味する．[RFC 1140]

実行可能イメージ(executable image)
オペレーティングシステム上で実行することのできるプログラム．プログラムはソースコードからコンパイルされてオブジェクトコードになり，次いでオブジェクトコードはリンクされて実行可能イメージが形成される．

指定ルータ(designated router)
DNAやOSPFのルーティングにおける概念．1つのブロードキャスト型回路(例えばイーサネット)は指定ルータをもち，末端ノードは経路の決定の必要なすべてのパケットをそれに送る．

集積(aggregate)
データのいくつもの列の操作を実行するのに使われる質問言語の機能．総計は集積の一例である．

集線装置(concentrator)
FDDIリング上のノードで，別の端末(station)にコネクションを提供する．802トークンリングではMAU(multistation access unit)として知られる．

終端(terminator)
イーサネットの両端にあって反射を防ぐ器具．

終端表現(termination expression)
Netwise RPC Toolで用いられる用語．データ構造がもち得るより少ないデータしかないときに終了表現が付加される．これによりRPC機構はデータ構造の使用していない部分に対して空のセルを送ることなく，真のデータだけ送ればよいようになる．

主幹ケーブル(trunk cable)
イーサネットケーブル(trunk)を，個々のノードに向かう付加的部分(トランシーバケーブル)と区別して指す言葉．

巡回冗長性検査(cyclic redundancy check)
伝送されるデータの集合から得られる数．受信側でCRCをもう一度計算し，伝送されてきたCRCの値と比較することにより，ある種の伝送誤りを検出することができる．

順序番号 (sequence number)
信頼できるトランスポート層サービスにより特定のコネクション上のすべてのパケットに付けられる一意番号．順序番号により，下位のネットワーク層やデータ層でパケットが失われたり，誤った順序で配達されたかどうかを，トランスポート層で知ることができる．

ジョイン (join)
関係データベース用語で，複数の表を組み合わせること．

状態機械 (state machine)
状態と呼ばれる論理的なサブタスクにプロセスを分割する仮想的な機械．状態は定義された順序に組織され，実行される．Netwiseはdispatch, client stub, server stubの中に，ソフトウェアによって状態機械を実装している．

証明機構(certification authority)
X.509やRSAの公開鍵認証システムで用いられる，ネットワークベースのソフトウェアプロセ

ス．ユーザの公開鍵を管理する．

初期化パケット(initialization packet)
トークンリングでノードがネットワークに加わるときに用いられるパケット．

ジョブキュー
印刷キューなどのように，貴重な資源への複数の要求を待ち行列に貯める方法．

シンボル(symbol)
ネットワークの MAC層で用いられる最小の信号要素．その後シンボルは物理層に送られて送信される．FDDIでは，1つのシンボルは物理層では5符号ビットである．

推奨プロトコル (Recommended Protocol)
Internet標準のプロトコル状態．システムはこれらのプロトコルを実装すべきであるが，必須ではない．

スキーマ(schema)
オブジェクトの構造やインスタンスの記述．[SunNet Manager]

スキーマファイル (schema file)
1つ以上のエージェントスキーマを含む Unix ファイル．エージェントスキーマファイルとも呼ばれる．[SunNet Manager]

スターカプラ (star coupler)
CIバスを使っているVAXクラスタにおいて，異なるノードを接続するのに使用される装置．

スタブ(stub)
RPCで用いられる小さなソースコード．スタブは被呼出しまたは呼出し手続きのように見えて，呼出しまたは被呼出し手続きからRPCの実装の詳細を覆い隠している．

ステーション管理 (station management)
FDDIリングの一部．アクティビティを監視し，全体の制御を行う部分．

ストリーム指向(stream-oriented)
トランスポートサービスの種類の1つで，クライアントが連続したストリーム状のにデータを送ることができるもの．トランスポートサービスは，送信されたすべてのデータが，同じ順序で重複なく，もう一方の端まで配達されることを保証する．信頼できるトランスポートサービスとして知られる．

ストリームヘッド(stream head)
ストリームへの入口で，STREAMS機構に接続された一連のソフトウェアモジュール．

スプール(spool)
高速な装置(例えばプログラム)が，低速な装置(例えばプリンタ)が後から処理を行うように，データを置いておく場所．

スロット (slot)
固定の大きさの表のエントリ．あるいは Local Area Transport アーキテクチャの副層で，複数のユーザが1つのデータグラムパケットを共有し，それぞれが別のスロットを使用することができる．

制御(control)
管理対象オブジェクトの特性を変化させるために，オブジェクトの属性値を変えられること．[SunNet Manager]

制約トークン (restricted token)
FDDIにおける非同期アクセスの特別なモードで，バンド幅は2人のユーザのextended dialogueだけに用いられる．

セッション（あるいはセション：session）
ネットワークを通じて交信している2つのプログラム間を流れる論理的データストリームを示すネットワーク用語．ネットワークの1つのノードからは，多くの場合，多数の異なるセッションが発していることに注意．

セマフォ(semaphore)
オペレーティングシステムの同期機構の1つ．

選出表現 (choice expression)
拡張されたC言語の表現で，RPCのコードが共用体のどの型をネットワークを通して送るべきかを指し示す．[Netwise RPC Tool]

選択的確認応答 (selective acknowledgment)
確立されたコネクション上で，拡張された確認応答情報を伝える TCP オプション．
具体的には，データの非連続ブロックが受信され，キューに入れられたことをデータ受信者が送信者に知らせるために送出される．データ受信者は順序番号空間中でこれらのブロックのすき間を埋めるデータを以後の再送により受信するのを待つ．このときデータ受信者は，普通TCPヘッダの確認応答番号フィールドのウィンドウの左端を1つ進めることでデータの確認応答を行う．[RFC 1072]

選択的プロトコル(Elective Protocol)
Internet標準のプロトコル状態．システムは，選択的プロトコルを実装してもしなくてもよい．一般的な考え方は，このようなことをするなら，この通り行えということである．[RFC 1140]

全二重(full-duplex)
通信回線の両側が同時に送信できることを表すデータ通信用語．同時には片側のみの送信が可能な半二重通信と対比される．

属性 (attribute)
エンティティに関して読んだり変更したりすることのできる"認知された性質"(perceived property)のこと．属性はネットワーク管理や名前づけサービスにおいて使用されている．オブジェクトユーザのパスワード属性がその一例である．関係データベースでは，属性は表の列に対する別名である．データ辞書 (data dictionary) やその他の情報モデルでは，属性は関係やエンティティに付加される．

属性グループ
名付けられた一連の属性．

属性名
エージェントスキーマによって定義される属性の名前．[SunNet Manager]

ソケット(socket)
プログラムへの入口．ユーザプログラムはソケットを使ってIPXと通信する．それぞれのプログラムは通常別々のソケットをもつ．

第4世代言語(forth-generation language)
IngresやOracleのようなデータベースパッケージとしばしばリンクされるいくつかの新しい言語．Fortranやその他の第3世代言語と対比される．

帯域幅
特定な通信回線を通して伝送することのできるデータの量．イーサネットは10Mbpsの帯域幅をもっている．

待機モニタ(standby monitor)
Token ringでの用語で，動作中のモニタが故障するのを待っていて，故障し次第乗り込もうと準備しているコンピュータを指す．副大統領のようなもの．

待機モニタ存在(Standby Monitor Present)
待機モニタから7秒ごとに送信されるパケットで，待機モニタの存在を知らせるのに用いられる．

代理エージェント(proxy agent)
マネージャに代わって管理を行うエージェントで，プロトコル族[suites]が違うなどの非互換性のため直接管理できないオブジェクトを管理するために用いられる．proxyと略されることもある．[SunNet Manager]

タグ数量詞(tag qualifier)
C言語の宣言文の拡張で，転送されるデータやパラメータのために用いられる．タグ数量詞はNetwise RPC コンパイラに対して，PDUを作成するときに，宣言された型の規則を使う代わりに，タグ数量詞の型の符号化規則を使用するようなRPCコードを作成するよう指示する．[Netwise RPC Tool]

多重化サーバ(multiplexing server)
バインディング要求の処理法の1つ．多重化サーバと呼ばれる単一のサーバがすべてのバインディング上のすべてのRPC要求を処理する．[Netwise RPC Tool]

多重処理サーバ(multiprocessing server)
バインド要求の処理法の1つで，サーバプロセスが与えられたバインディング上のすべてのRPC要求を処理する役割をもった新しいプロセスを起動するもの．[Netwise RPC Tool]

タプル(tuple)
関係データベースで用いられる用語．タプルはファイル管理システムにおけるレコードに相当し，表の中の1列分のデータに対応する．

探索パス(search path)
DOS，Unix 他のオペレーティングシステムの機構で，コマンドがどこのディレクトリにあるかユーザが知らなくても指定できる．オペレーティングシステムは search path 中のそれぞれのディレクトリを調べてコマンドを探し，ファイルを見つけ出す．

端末エミュレータ(terminal emulator)
コンピュータが端末をエミュレートできるようにするプログラム．ワークステーションがホストから端末のように見えるようになる．

チェーン呼出し(chained call)
RPCで呼び出された遠隔手続きの処理中に入れ子になって行われるRPC．最初のRPCをparent call，入れ子になった内側のRPCをchild callという．[Netwise RPC Tool]

チャネル(channel)
370アーキテクチャに直接接続される高速の接続をいうIBMの用語．通信コントローラに接続される通常のデバイスが56kbpsで通信するのに対し，"チャネル接続"装置は最高3Mbpsで通信できる．

抽象データ型(abstract data type)
プログラミングやデータベースシステムで用いられている概念．抽象データ型は，それを扱う

ときの内部の詳細が隠されているようなユーザ定義のデータ型である．例えば，日付，という抽象データ型は内部では整数として格納されるが，外部では日付特有の記述や操作がなされる．

中間システム(Intermediate System)
エンドシステムの間の経路を提供するために，ルーチングおよび中継機能を行うOSIシステム．中間システムはネットワーク層より上の機能はもたない(実際にはOSIルータの現実的な実装においては，少なくともネットワーク管理機能のため，ある程度のエンドシステムとしての機能をもつであろう)．基本的にはInternetルータと等価．[RFC 1136]

ディスパッチ関数(disppatch function)
エージェントのルーチンで，要求(request)に応えるため適切なエージェント関数(function)を呼び出すもの．[SunNet Manager]

ディスパッチャ(dispatcher)
Netwise RPC Compilerによりサーバソースコードの一部分として生成される手続き．ディスパッチャはクライアントの手続きから要求を受け取り，その要求を適切なサーバスタブに振り向ける．[Netwise RPC Tool]

ディレクタ(director)
ディレクティブ(エージェントにある機能を実行させるコマンド)を発行する管理主体．SunNetマネージャはディレクタの一例．

ディレクトリアクセスプロトコル(Directory Access Protocol)
ディレクトリユーザエージェントとディレクトリシステムエージェントの間の通信に使われるX.500プロトコル．

ディレクトリシステムエージェント(Directory System Agent)
Directory Information Baseへのアクセスを管理，提供するソフトウェアを指すX.500用語．

ディレクトリシステムプロトコル(Directory System Protocol)
ディレクトリシステムエージェント間の通信に用いられるX.500プロトコル．

ディレクトリ情報木(Directory Information Tree)
Directory Information Baseに含まれるオブジェクトの階層的な名前空間を指すX.500用語．

ディレクトリ情報ベース(Directory Information Base)
アプリケーションエンティティ，人，端末，配布リスト，その他のオブジェクトを把握しておくために使われる分散データベースを指すX.500用語．

ディレクトリマネージメント領域(Directory Management Domain)
Directory Information Baseのある部分の管理を担当する管理グループ．directory management domainにはディレクトリシステムエージェントが少なくとも1つと，ディレクトリユーザエージェントが0またはそれ以上含まれている．

ディレクトリユーザエージェント(Directory User Agent)
ユーザにサービスを提供するソフトウェアを指すX.500用語．ディレクトリユーザエージェントはOSI ROSEプロトコルを使って1つ以上のディレクトリシステムエージェントと通信する．

デカルト積(cartesian product)
2つのデータのリストに対する，すべての可能な組合せの集合．

データアクセスプロトコル(Data Access Protocol)
Digital Network Architecture の第6層で用いられるプロトコル．ネットワーク上の2ノード間でデータをやりとりするための機能を多数提供する．File Access Listener も参照．

データ国別コード(data country code)
地理的位置に基づく，ISOにより管理されている一意OSIアドレスの形式．コードはISO 3166で定義されている．International Code Designatorも参照．

データグラム(datagram)
2つのインターネットモジュールの間で転送される単位．Internetプロトコルは，データグラムと呼ばれる一塊のデータを発信元から宛先へ転送する方法を提供する．Internetプロトコルは信頼できる通信機能を提供せず，end-to-endでも hop-by-hop でも確認(acknowledgements)は行わない．データの誤り制御も行われず，ヘッダのチェックサムだけである．再送も行われないし，フロー制御もない．IP の項目も参照せよ．[EFC227]

データ終端装置(Data terminal equipment)
ネットワークに対するDCEインターフェイスに対して，ユーザ装置へのインターフェイスを表すX.25用語．

データ報告(data report)
オブジェクトの属性のうち解析に対して報告さ

れたものを，イベント報告に対比していう．[SunNet Manager]

デッドロック(deadlock)
並行実行環境で用いられる用語．あるユーザが資源をロックした後で，他の資源が解放されるのを待っているとき，第2のユーザがそれと逆の状況にある場合をいう．デッドロックは任意に一方のユーザを選んで，それがもっているロックを解除することで解消する．デッドロックは死の抱擁(deadly embrace)ともいわれる．

デフォルト経路 (default route)
ルーティングテーブルに明示されていないネットワーク番号に宛てられたデータをすべてふり向ける先として使われるルーティングテーブルのエントリ[RFC1177].

デーモン(daemon)
特定のユーザに対応するのではなく，メールデーモンのようにサービスを実現するプロセスを表すUNIX用語．VMSではdetached processという．

デュアルポート化(dual-porting)
2つの異なるコンピュータがディスクドライブを共用できるようにすること．

電子メール(electronic mail)
ネットワークを経由してユーザがメッセージを交換することを可能にする一連のプログラム．

伝送制御プロトコル(Transmission Control Protocol)
TCP/IPにおいてデータの確実な配達のために用いられるトランスポートプロトコル．

伝播速度(propagation velocity)
信号が電線上を伝播する速さ．信号は，池に小石を投げ込んだときのさざ波のように電線上を伝わる．

同期(synchronization)
複数のユーザのタスク間の協調．同期メカニズムとしては，ロックやセマフォが挙げられる．

同期式 (synchronous)
FDDIのサービスクラスで，各々の要求者は，あらかじめ割り当てられた最大のバンド幅を得るので，応答時間が保証される．

同期処理(synchronous processing)
他の手続きを呼んだ後，呼出し手続きが中断され，応答を待つような一般的な動作方法．

同軸(coax,coaxial)
ケーブルの一種類．IBM 3270端末やベースバンド，および細径イーサネット(ThinWire Ethernet)で用いられる．

トークンバス(token bus)
トークンリングやイーサネットに代わるローカルエリアネットワークの1つ．MAPプロトコルで用いられている．トークンバスはマルチプルアクセスプロトコルを用いているがトークンを“所有している”装置だけが送信を行うことができる．

トークンリング(token ring)
コンピュータがリング状に接続されるローカルエリアネットワーク．ノードはリング上をトークンが回って来るのを待ち，トークンを受け取ると送信することができる．送信が終わればトークンを開放して次のノードに渡す．FDDIを参照．

年寄りパケット(aged packet)
最大通過数を越えたルーティング層パケット．

トポロジー(topology)
ネットワークトポロジーとはコンピュータとそれらの間の回線を表すものである．ネットワーク層がパケットを最終目的地に送るためには現在のネットワークトポロジーを把握していなければならない．

ドメイン(domain)
ある特定のサービスが実現される領域のこと．メッセージドメインとは，そのドメインのメッセージ転送エージェントがメッセージを配達できる範囲のこと．

ドメイン固有部分(Domain Specific Part)
OSIアドレスのうち各部署で(locally)割り当てられる部分．

ドメイン名前システム(Domain Name System)
Internetでホストコンピュータの名前をアドレスに変換するために使われている機構．DNSでは直接Internet上にないホストコンピュータについても同じ形式の登録名をもつことが可能である．

トランザクション(transaction)
全体として論理的な一体をなすような一連の操作．トランザクション全体が実行されるか何も実行されないかのいずれかでなければならない．

トランシーバ(tranceiver)
イーサネットのネットワークで用いられる用語．トランシーバはイーサネットの媒体(多くの場合は同軸ケーブル)に接続されるハードウェアである．次いでトランシーバとホストシ

ステムのイーサネットコントローラが接続される．

トランシーバケーブル(tranceiver cable)
トランシーバとイーサネットコントローラの間を結ぶ50m以内のケーブル．

トランスポートコネクション(transport connection)
トランスポート層によって提供される，2つのユーザの間の仮想通信チャネル．

トランスポートサービスアクセスポイント(transport service access point)
あるエンドユーザから見たときのトランスポートモジュールへのエントリポイント．

トランスポートレベルインターフェイス(Transport Level Interface)
トランスポート層と上位層ユーザの間のインターフェイスに関するAT&Tにより開発された仕様．

名前空間(namespace)
一般的に配布されている名前の集合．その中ですべての名前は一意である．

名前空間生成タイムスタンプ(Namespace Creation Timestamp)
他と区別するための，名前空間の一意のID．

名前結合プロトコル(Name Binding Protocol)
論理名をホストアドレスに対応させるためのAppletalkのプロトコル．

名前つきパイプ(Named Pipes)
全二重の通信パスを維持するのを可能にするためのプロセス間のプロトコル．パイプとは通信の終端であり，プロセスはパイプを通じて各機能へのエントリを得る．パイプの名前は，ネットワーク内で維持・登録され，パイプがサービスにアクセスするのを可能にしている．

ニブル(nibble)
半バイトのこと．

入出力(input/output)
主記憶とディスク駆動装置，端末，プリンタ，その他の装置との間のデータの転送を表す一般的用語．

認可(authorization)
ある人あるいはプロセスが，ある特定の行為をなすことができるかどうかを判断すること．認証と対比される．

認証 (authentication)
ある人やプロセスの身分を確認する機能．

ネットワーク(Network)
単一のInternetネットワークのこと(サブネットに分かれていたり，分かれていなかったりする)．[RFC 950]

ネットワークアドレス(network address)
ユーザが存在しているネットワークの番号．インターネットワークの各ネットワーク(データリンク)は番号を割り当てられている．ステーションの完全なアドレスは，ネットワークアドレスにそのネットワーク内でのノードのローカルアドレスを加えたものである．

ネットワークサービスアクセスポイント (Network Service Access Point)
OSIにおいてエンドシステム中で広域的にアクセスできる(そしてアドレスは広域的に一意である)ネットワーク層/トランスポート層の境界上にある概念的な点．1つのNSAPはネットワーク層の上にあるサービス(トランスポートプロトコルなど)を表す．1つのエンドシステムは複数のNSAPをもつことができる．NSAPアドレスはおおよそInternetにおける[アドレス，プロトコル]対と等価である．[RFC 1136]

ネットワークサービスプロトコル (Network Service Protocol)
DECnetのトランスポート層プロトコル．

ネットワーク情報サービス(Network Information Service)
SunのOpen Network Computing (ONC)族の名前サービスである．Yellow Pagesも参照せよ．

ネットワークセレクタ(nework selector)
第N層の特定のユーザを選定する第N層におけるアドレスの一部．いいかえると，次の層のアドレス．

ネットワーク番号(network number)
着目しているノードが属しているネットワークを示す，インターネットアドレスの一部．

ネットワークファイルシステム(Network File System)
Sun Microsystems社によって開発された分散ファイルシステムであり，TCP/IPシステムで広く利用されている．

ネットワークライブラリ(network library)
Netwise RPC Toolについてくるソフトウェアモジュール．ネットワークライブラリは，完全なプログラムを形成するために，RPCコンパイラによって生成されるコードに付加される．このライブラリは，ネットワーク通信へのアクセスの詳細をプログラムの残りの部分に見えなくする．

ノード(node)
集合の中の個々の要素．例えば，イーサネットノードとは，トランシーバによってケーブルに接続されたデバイスのことであり，リピータやブリッジやコンピュータが含まれる．また，ファイルシステムノードとは，ディレクトリや個々のファイルのことである．

媒体アクセス制御 (Medium Access Control)
ISOデータリンク層の下半分．論理リンク制御の項目も参照せよ．

パイプ(pipe)
名前つきパイプ(Named Pipes)参照．

配列(array)
プログラミングで使われるデータ型．

バケット(bucket)
入出力の転送時に用いられる，ファイルの仮想的なブロックの集まり．

パケット(packet)
パケット交換ネットワーク中を伝送されるデータの単位．この用語はそれほど厳密ではなく使われる．あるInternetの文献では，物理的なネットワーク上を送られるデータのことに限定して用いるのに対し，別の文献ではInternetをパケット交換ネットワークとして見て，IPデータグラムをパケットと記述しているのもある．[RFC1177]

パケット組立て/分解装置 (Packet Assembler/Disassembler)
X.25ネットワーク上で，非同期トラフィックをパケットに格納することで，非同期端末から同期網のX.25ネットワークを使うことを可能にする専用目的のコンピュータ．

パケット交換(packet switching)
データをパケットにパッケージ化するネットワーク．コンピュータは専用のコネクション(回線交換として知られている)の場合よりも，パケットの方が多くの仮想コネクションを扱うことができる．パケット交換は，X.25および多くのネットワーク層プロトコルの基礎をなしている．

パケット交換プロトコル(Packet Exchange Protocol)
各パケットが別々に確認されることを要求するXNSのトランスポートプロトコル．PEPに似たプロトコルは，Netware Coreプロトコルの基礎となっている．

パケットバッファリング方式 (packet buffering style)
普通，PDU全体が同時にメモリ上には存在しないようなバッファリング方式．PDUの各々の部分は，比較的小さな固定長のバッファ中で連続的にパックされたり，アンパックされたりする．この方式により，PDU転送とパッキングやアンパッキングを並列に行うことができる．[Netwise RPC Tool]

バス(bus)
計算機内で，周辺装置とCPUやメモリとをつないで通信を行うことができるようにする部分．IBMのMicro Channel Architectureは周辺バスアーキテクチャの例である．また，イーサネットやトークンバスのように，ポイント対ポイントネットワークでなく，マルチプルアクセスの機能をもつネットワークのこともいう．

パス(path)
ファイルシステムの概念として，パスはファイルがどのフォルダまたはサブディレクトリ群の中にあるのかを示している．ネットワークの観点では，パスはパケットが出発地から目的地まで通る経路のことである．パスは一連のデータリンクまたはホップである．

バースト性トラフィック(bursty traffic)
データ通信の用語で，一様でないデータ伝送パターン．

パック(Pack)
遠隔手続き呼出しにおいてマシン依存の形式から，マシンに依存しない形式にデータを書き換えることを表す用語．

バックアップ
元の記憶装置(通常ディスクドライブのこと)が破損した(corrupted)ときのために，格納されている情報を複写しておくことである．鉛管工事や下水道などにおいてよく使われる意味である"オーバーフローさせること"と対比せよ．

パック手続き(pack procedure)
Netwise RPCコンパイラによってつくられる手続きで，1つないしそれ以上のデータ型を，ネットワーク中を伝達するための標準フォーマットのPDUに符号化する．[Netwise RPC Tool]

バックボーン
ネットワークで使われる用語で，異なるフロアや部門を連結するケーブルのこと．部門ネットワークや職場ネットワーク(work area network)

と対比される．

発信元アドレス(source address)
ネットワーク上のデータパケットの発信元．

バッチ呼出し形式 (batched calling style)
無応答呼出し形式の特別な場合で，要求の伝送や配達がすぐに行われるのでなく，以後の要求といっしょにすることのできる形式．[Netwise RPC Tool]

バッファ(buffer)
データを保持するのに用いられる計算機の主記憶の一部．

バッファリング方式(buffering style)
データをPDUに符号化したり，PDUからデータを復号化したりする際に，メモリ上のバッファをどのように管理するかを指定する方法．RPC Toolでは，packetとmixedという2つのバッファリング方式をサポートしている．[Netwise RPC Tool]

ハブ(hub)
他のいくつもの装置と接続される装置．ARCnetでは，いくつものコンピュータをつなぎ合わせるのにハブが用いられる．メッセージ処理システムではネットワーク経由でメッセージを転送するのにハブが用いられる．

パブリックドメイン(public domain)
使用料を払わずに利用することができる知的財産．大学で開発されたコンピュータソフトウェアの大部分はパブリックドメインである．

パーマネントバーチャルサーキット(parmanent virtual circuit)
電話ネットワークにおける専用線のように，ずっと保持されている回線．

バラン (balun)
Balanced/unbalanced. 2つの物理的な媒体の間にあるアダプタで，インピーダンスの差を整合するためのもの*.
(*訳注：インピーダンスの差を吸収することよりも，平衡回線と非平衡回線の間の変換をすることが名前の由来である．)

ハンドル(handle)
ネットワーク上で外部からアクセスされるファイルやディレクトリを指すのに用いられる数．

半二重(half-duplex)
全二重(full-duplex)参照．

非永続バインディング(nonpersistent binding)
遠隔手続きを呼び出すたびにコネクションの確立と切断を行う，遠隔手続き呼出しにおけるバインディング方式．

非永続方式(nonpersistent style)
バインディングを管理する方法の1つ．遠隔手続き呼出しの宣言でこの方式が用いられると，Netwise RPC Compilerはその遠隔手続きに対して非永続バインディングをサポートするためのコードを生成する．[Netwise RPC Tool]

被管理オブジェクト (managed object)
管理の対象となる特定の資源(計算機システム，ネットワークインターフェイスなど)を指す．コンソールの中では要素とも呼ばれる．[SunNet Manager]

引数PDU
要求PDUのユーザデータの部分をもっているPDU．[Netwise RPC Tool]

ピギーバック(piggybacked)
上に付け加えること．以前のパケットに対する確認応答を必要とするプロトコルで用いられる用語．確認応答は，しばしばその方向に向けられたデータと同じパケットの中に積み込まれる．

ビーコン(beacon)
トークンリングに重大な故障が起こったことを通知をするパケット．

非推奨プロトコル(Not Recommended Protocol)
Internet標準のプロトコルの状態．これらのプロトコルは，一般的な用途には勧められない．これは，これらのプロトコルの限られた機能，特殊な性質，実験的または歴史的な状態によるものであろう．[RFC 1140]

非制約トークン(nonrestricted token)
非同期モードで全ユーザが利用可能なFDDIトークン．

ビッグエンディアン (big endian)
複数オクテットのデータを格納するとき，最もアドレスの小さいオクテットを最上位桁 (most significant) とする計算機．エンディアンとリトルエンディアンも参照．

必須プロトコル(Required Protocol)
Internet標準のプロトコル状態．システムは必須プロトコルを実装しなければならない．

ビットマップ(bit mapped)
グラフィックスの用語で，ディスプレイのすべてのビットを制御できることをいう．キャラクタ指向端末(character-oriented terminal)に対して使われる．

非同期(asynchronous)
FDDIにおいてデータ伝送方式を指す用語であり，すべての要求がリングの帯域幅を競争する方式である．

非同期イベント
ユーザが次に起こるイベントを予測できないように，システムに非同期的に発生するイベント．

非同期処理
呼び出した手続きの実行が，他の手続きの呼出し後その応答待ちによって中断されないような動作方式．同期処理を参照．[Netwise RPC Tool]

非同期通信
すべてのバイトを個別に送る通信方式．同期通信方式(例えばX.25やイーサネット)は，いくつものデータをフレームやパケットに束ねて送る．

非同期平衡モード (asynchronous balanced mode)
IEEE 802.2標準で使われている用語であり，コネクションの片側だけが送信できる状況を指している．

非排他的ロック(nonexclusive lock)
他のユーザからの情報の読出しは許すが，書込み操作は許さないという種類のファイルのロック．

ヒープ(heap)
データが置かれる順番が決まっていない記憶構造で，取出しごとに表全体の走査を必要とする．

ビュー(view)
コンソールのMDB中の構成要素の集まり．ビューの構成要素は要素(element)と呼ばれる．ビュー自体も要素であるので，他のビューの構成要素であり得る．要素は複数のビューの構成要素であってもよい．[SunNet Manager]
また，データベース用語で，1つ以上の表の集まりを単一のデータベースの表として見えるようにすることをいう．

表 (table)
関係データベースの用語で，行および列に区分されたデータ．

表現マクロ(expression macro)
Netwise RPCコンパイラが受け付けるC言語の拡張で，ポインタマクロと指標マクロの2種類からなる．表現マクロは選択(choice)表現および終端(termination)表現で用いることができる．RPCマクロ参照．[Netwise RPC Tool]

標準プロトコル(standard protocol)
Internet標準の区分．IABはInternetの公式な標準プロトコルとして制定した．これは2つに分類できる:(1)IP プロトコルとその上位．Internet全体に適用できるプロトコル．(2)ネットワークに固有なプロトコル．一般に特定の種類のネットワーク上でIPをいかに動作させるかについての仕様．

非ルーチングノード(nonrouting node)
エンドノード(end node)を見よ．

ビンデリ(bindery)
ユーザパスワードなどの管理情報の保存に用いられている，NetWareオペレーティングシステム上のファイル．

ピンポン(ping-pong)
各々のパケットが，個々に確認されることを必要とするトランスポートプロトコルの種類．ノードが新しいパケットを送るには，その前に確認応答を待たなければならない．PEPはこの一例で，1ウィンドウ分の確認応答の得られていないパケットを許すSPXと対比される．

ファイバ分散データインターフェイス(Fiber Distributed Data Interface)
トークンリングに基づく100Mbpsの光ファイバLAN(fiber optic LAN)の標準．

ファイルキャッシュ(file cache)
ファイルの一部分を主記憶上に保持すること．計算機がデータを要求し，そのデータがキャッシュ上にある場合，はるかに遅いディスクドライブからそのデータを再び取ってくる必要がなくなる．

ファイルシステム(file system)
データページ(pages of data)をディスク上に蓄えたり，ディスクから取り出したりする責任を担っているオペレーティングシステムの一部分．

ファインダ(finder)
Macintoshファインダ参照．

ファックス(fax)
ファクシミリ(facsimile)．ダイヤルアップ(dial-up)電話回線を介して1インチ当り200ドットのビットマップを伝えることによるメッセージ転送サービス．

フィールド(field)
データベースアプリケーションのような書式に基づくシステムを設計する際に用いられる用

語．1つのフィールドはデータ入力に用いられるスクリーンの一部分であり，変数に自動的にマップされる．フィールドはリバースビデオ(reverse video)やデフォルト値のような属性をもつことができる．

フォールトトレランス(fault tolerance)
実行継続中のハードウェアおよびソフトウェアの故障(failure)に対する許容度を表す(reflect)計算機システムの属性．

付加価値再販売者（Value-added reseller）
他の企業の製品をより洗練された製品の中に埋め込む企業．

付加価値網（Value-Added Network）
ネットワーク上のネットワーク．

不均質(heterogeneous)
違っていること．

不均質ネットワーク(heterogeneous network)
異なるネットワークプロトコルや異なる種類のコンピュータから構成されているネットワーク．SNAゲートウェイを用いてSNAプロトコルととDNAプロトコルの間を接続するネットワークは不均質ネットワークの例である．

符号なし（unsigned）
正の数値だけをもつデータ型．

フック(hooks)
プログラミング技術．プログラマが既存のコードに新しいコードを付け加えることを可能にするもの．このとき，既存のプログラムは付加されたコードを実行するフックをもつという．

浮動小数点(floating point)
ほとんどのオペレーティングシステム上で元からある(native)データ型．浮動小数(floating point number)とは小数点の後に数をもつことができる数のことで，これができない整数と対比される*．
(*訳注：正しくは数を有効数字部分と指数部分で表すこと．)

太径ケーブル（Thick Wire）
イーサネットおよび同軸ケーブルの10BASE5規格の別名．

ブリッジ(bridge)
2つのイーサネットネットワークをつないで1つの拡張イーサネットにするための装置．ブリッジは他のネットワークへ宛てられたパケットだけをネットワークをまたいで中継するだけである．Novellでは，ネットワーク層でパケットを受け取り，他のネットワークに転送するようなコンピュータを指す．

プリミティブ(primitive)
ISOでの定義．あるエンティティによって，別のエンティティに提供されるサービスの要素．

プリンタアクセスプロトコル(Printer Access Protocol)
プリンタにアクセスするためのAppleTalkのプロトコル．

プリントスプーラ(print spooler)
印刷のためのジョブを，一度にいくつも受け付けるソフトウェアプログラム．スプーラはプリンタへのアクセスを制御し，入ってくるジョブを実行のための待ち行列に入れる．

フルネーム(full name)
名前空間上で唯一であいまいさのない名前．

フレーム(frame)
ヘッダを付けてカプセル化されたバイトデータの並び．データリンク層はデータフレームを送受する．“フレーム”はしばしば“パケット”と同じものとして用いられる．もっとも，技術的にはパケットはプロトコルスタックのネットワーク層からのデータを指す．したがって，通常パケットはフレームの中に格納される．

プレゼンテーション構文(presentation syntax)
異機種の環境でデータを表現する標準方式．The Abstract Syntax Notation 1(ANS.1)はプレゼンテーション構文の一例である．

プレゼンテーションマネージャ(Presentation Manager)
ユーザインターフェイスを管理するOS/2オペレーティングシステムの構成要素．

プログラマ用階層的対話型グラフィックシステム(Programmer's Hierarchical Interactive Graphics System)
隠面除去や陰影など，高度な機能をもった描画システム．

プロセス間通信 (interprocess communication)
2つのプロセスが，パラメータを渡したり値を返したりして通信すること．遠隔呼出しはプロセス間通信の特別な場合である．

ブロッキング(blocking)
特定の条件が満たされるまでアプリケーションプロセスの実行を中断すること．同期処理で起こる．クライアントアプリケーションが遠隔手続き呼出しが終わるまで実行を中断することを指すのによく使われる．[Netwise RPC Tool]

ブロッキング呼出し(blocking call)
手続きから戻るまで呼出し側が実行を中断するような手続き呼出し.

ブロック(block)
コンピュータが入出力を行う際の単位. 512バイトから8キロバイトの間であることが多い.

ブロードキャスト(broadcast)
あるサービスのすべてのユーザに対して情報を送ること. 例えばイーサネットにおけるブロードキャストでは, ネットワーク上のすべてのアドレスにイーサネットのパケットが送られる*. (*訳注：実際はある特定のアドレス(全ビット"1")がブロードキャストのために予約されており, 各ノードが自分のアドレスの他にブロードキャストのためのアドレスを認識する.)

ブロードキャスト呼出し方式(broadcast calling style)
要求PDUをユーザが指定したいくつかのサーバのすべてに送るようなRPCの呼出し方式. クライアント側のスタブ状態機械を, 複数の応答を待つように改造することもできる. [Netwise RPC Tool]

ブロードキャスト隣接ルータ(broadcast router adjacency)
あるノードからブロードキャストが可能な回路に接続されているルータ.

プロトコル(protocol)
2台のコンピュータがメッセージを交換する際に従うべきメッセージの形式および規則の形式的な記述. プロトコルは低レベルなマシン間インターフェイスの詳細を記述することもでき(例えば信号線を通してビットやバイトが送られる順序), また高レベルな割当てプログラム間のデータ交換の仕方を示すこともできる(例えば2つのプログラム間でInternetを経由してファイルを送る方法). [RFC 1177]

プロトコル階層(protocol stack)
プロトコル階層の各層にはそれぞれ機能があり, これらが全体としてネットワークサービスを形成している. プロトコル階層の各層は, それぞれ下位のサービスモジュールを用い, この上にサービスを形成する.

プロトコルデータ単位(protocol data unit)
ある層は同位の層とパケットを送り合うことにより通信を行う. 各パケットには, アドレスや確認応答要求など同位の層が扱うべき情報がヘッダとして含まれている. また, パケットにはデータであるプロトコルデータ単位も含まれており, その層のクライアントへそのまま渡される.

ブロードバンド(broadband)
ケーブルTVに似たアナログメディアで, 多数のネットワークのトラフィックを周波数多重化して伝送する. 広帯域で距離がのばせるという特徴のため, 構内回線に適している. イーサネットやトークンバスなどのいろいろなデータリンクプロトコルが用いられる.

フロントエンド(front end)
ユーザと対話するプログラム. フロントエンドはバックエンドに要求を送ることでデータを得る.

分散アプリケーション(distributed application)
アプリケーションのうち, そのコードがネットワーク上の複数のアプリケーションプロセスによって実行されるものをいう. [Netwise RPC Tool]

分散データベース(distributed database)
ユーザには1つのデータベースに見えるが, 実際はいくつかの別々のデータ格納場所(repository)から構成されているもの.

分散名前づけサービス(distributed naming service)
ネットワークに基づくサービスで, ある資源(resource; プリンタやファイルシステムなど)の現在のアドレスをユーザが見い出すことができるようにするもの.

分散ファイルサービス(Distributed File Service)
ネットワーク上のファイルをすべて自分のマシンにある(local)かのように見せるDECの製品. ネットワークファイルシステムと同様のもの.

分離可能呼出し形式 (separable calling style)
クライアントスタブ状態機械が遠隔手続き要求処理処理と応答処理を分離する呼出し形式. 非同期処理の前提条件. [Netwise RPC Tool]

米国標準化機構(American National Standards Institute)
いくつかの米国の標準作成を調整する私的組織. 国際標準化機構に対して合衆国を代表する.

冪(べき)等(idempotent)
操作を2度以上行っても結果が同一であること. 例えばXを25にすることは冪等である. Xに25を加えることは冪等でない.

冪等呼出し形式(idempotent calling style)
クライアントスタブの状態機械が返答を受信す

るまで要求を繰り返すような呼出し形式．その結果として，遠隔手続きが2度以上実行される可能性があることに注意．[Netwise RPC Tool]

ページング(paging)
仮想メモリオペレーティングシステムにおけるメモリ管理手法の1つ．プログラムのほんの一部(2，3ページ)だけが実際にメモリ上にある．新しい部分が必要となると，メモリへとページングされる．

ベースバンド
イーサネットの同軸ケーブルによる実現．ThickWireとも呼ばれている．

ヘッダ(header)
パケット中で実際のデータに先立つ部分で，発信元および宛先のアドレスや誤り検査のためのフィールドが含まれている．

別名(alias)
他の名前に翻訳される名前．DNSソフトリンクは別名の一例である．

変数モード(variable mode)
外部変数モード(external variable mode)を参照．

ボー(baud)
古い(遅い)モデムで使われている用語で，アナログ信号の各々の変調を意味する．300ボーの信号は1秒間に300回変調する．1回の変調で何ビットか伝送することができるので，より高速のモデムにおいてより正確な呼び方はbpsである．

ポインタテーブル(Pointer Table)
Netwise RPCコードによって作る表で，PDUに符号化されるべき個々の構造体へのポインタを格納している．この表は構造体の複製を避けたり，遠隔手続きに対して環状あるいは自己参照型の構造体を渡すことを可能にするために，RPCコードによって用いられている．[Netwise RPC Tool]

ポインタマクロ(pointer macro)
変数の要素，構造体変数，または構造体変数のメンバへのポインタを表す表現マクロ．ポインタマクロは$記号で始まる．[Netwise RPC Tool]

報告(report)
属性値やエラーメッセージが含まれている回答(通常はエージェントからマネージャに送られる)．応答(response)と呼ばれることもある．

捕獲(capture)
トークンリングネットワークで，リングからトークンを除去する動作．

ホスト(host)
ユーザに対してサービスを直接提供するコンピュータ(視界の外にある専用サーバと対比される)．

ホスト番号(host number)
Internetアドレス中で，(サブ)ネットワーク上のどのホストを指しているかを表す部分．[RFC 1177]

ホストフィールド(Host Field)
Internetアドレスにおいて，特定のホストを表すのに用いられるビットフィールド．[RFC 950]

細径ケーブル(Thin Wire)
イーサネットネットワークで用いられる細くて安価なベースバンド同軸ケーブル．CheaperNetとも呼ばれる．

ポータル(portal; 玄関)
イーサネットサービスではいくつかの異なるユーザがいることがあり，それらはそれぞれポータル，すなわち認識番号をもっている．外から到着したパケットは，適切なポータルへと配送される．

ホップ(hop)
ルーチングで用いられる用語．1ホップは1つのデータリンクである．ネットワーク上の最終的な宛先までの経路は，発信元から発する一連のホップである．最小コストの経路を計算できるように，各々のホップにはコストが対応づけられている．

マウス(mouse)
ワークステーションで用いられる指示装置．またはげっ歯類に属する小動物．

マウント(mount)
あるファイルシステムを，局所的なファイルシステムで使えるようにする過程．mountシステムコールにより，カーネルに新しいファイルシステムについて知らせる．ファイルが遠隔コンピュータ上のものである場合には，NFSのマウントプロトコルが用いられる．

マーシャリング (marshalling)
RPC呼出しで使われる用語．自ホスト内の手続き呼出しを，ネットワークを通じて送れるような形式にすること．

マネージャ (manager)
ユーザが特定のネットワークや装置を管理するのを助けるために，ユーザに代わって仕事をす

るプログラムやプロセスの総称．通常エージェントに要求を出して情報を集めさせ，それを手に入れてユーザのために画面に表示する．マネージャアプリケーションあるいはマネージャプロセスとも呼ばれる．[SunNet Manager]

マネージャサービス (Manager Services)
エージェントとマネージャが通信できるようにするための関数ライブラリをマネージャ側から見たもの．[SunNet Manager]

マネージャステーション (manager station)
あるネットワークや装置を管理する人の使っているワークステーション．SunNet Manager Console(SNM)のような管理アプリケーションが走っているのが普通である．[SunNet Manager]

マルチキャスティング(multicasting)
イーサネットのアドレッシングで用いられる用語．マルチキャストアドレスとは，イーサネット上のユーザの一部分に宛てられたアドレスのことである．LATの各ノードは，現在の自分の状態に関する情報を交換するのに，マルチキャストアドレスを用いて行っている．そのイーサネットのすべてのユーザが受信するアドレスであるブロードキャストアドレスと対比される．

マルチキャスト(multicast)
複数のノードが応答するアドレス．ネットワークのすべてのノードが応答するブロードキャストと対比される．

マルチステーションアクセスユニット(multistation access unit
トークンリングで用いられる装置で，複数のステーションをリングに接続するのに使う．イーサネットのマルチポートトランシーバに似ている．

マルチステートメントトランザクション (multistatement transaction)
あるデータベースに対する複数の異なる操作が，1つのトランザクションにまとめられたもの．ユーザの中止命令やシステムの障害によりこれらの操作のうちどれか1つでも実行されなかった場合，トランザクション全体がもとに戻される．マルチステートメントトランザクションでは，すべての操作が実行されるか，どの操作も実行されないかのいずれかである．

マルチスレッド(multithreaded)
1つのプロセスが複数の実行スレッドをもつことができるというオペレーティングシステムの機能．各スレッドは親プロセスの制御の下に置かれる．例えばOS/2などにおいて実現されている．

マルチスレッドサーバ (multithreading server)
バインド要求に対処する方法の1つで，1つのサーバプロセスが与えられたバインディング上のすべてのRPC要求を処理するための新しいスレッドを作るもの．[Netwise RPC Tool]

マルチセグメントイーサネット (multisegment Ethernet)
複数のイーサネットセグメントをリピータを用いて接続し，一体化したもの．マルチセグメントイーサネットでは，あらゆる信号がすべてのノードに伝わる．反対側のイーサネット宛のパケットしか中継しないMAC-layer bridgeを用いた拡張イーサネットと対比される．

マルチポイント(multipoint)
データリンク層における概念で，複数のノードが1つの物理的な通信媒体を共有していることを指す．マルチポイント状況では，1つのノードがその通信線のコントローラとなり，所属するすべてのノードに定期的に問合せを行って，データの送信を望んでいるかどうかを調べる．イーサネットのような，各ノードが許可なしで送信できるマルチアクセス通信媒体と対比される．

マルチポートトランシーバ(multiport transceiver)
複数のイーサネットトランシーバを1つの装置にしたもの．1本のケーブルに対する集線装置として使うこともできるし，それ自体を独立したイーサネット(“缶入りイーサネット”として知られている)として使うこともできる．

マルチポートリピータ(multiport repeater)
イーサネットのリピータの一種で，複数のEhternetセグメントを接続して，マルチセグメントイーサネットを構成する．典型的にはThinWireネットワークで用いられる．

マルチリンクエンドノード(multilink end node)
ネットワークに対して複数の接続をもつエンドノードのこと．このようなエンドノードはデータの送受信をどのリンクを用いても行えるが，他のノードのために経路制御を行うことはしない．

無応答呼出し方式(no-reply calling style)
サーバスタブ状態機械が応答を生成せず，クライアントスタブ状態機械が応答を待たない呼出し方式のこと．[Netwise RPC Tool]

メッセージ完全性検査指標(Message Integrity Check: MIC)
メッセージの中身から導かれる値．メッセージと共に送られ，メッセージの中身が変化していないことを確かめるのに使う．MICはメッセージをわずかに変えても大きく異なる結果を与えるという特徴がある．

メッセージ処理システム(message handling system: MHS)
例えばX.400のように，電子メールのようなメッセージをやりとりするための体系化されたプロトコルのこと．

メッセージ伝搬エージェント(message transfer agent: MTA)
ネットワークの構成員のうちメッセージの転送を受け持つものの集合を指すX.400の用語．最終のMTAが，メッセージの読出しや編集などのエンドユーザとの対話を行うユーザエージェントにメッセージを配達する．

メッセージルータ (Message Router)
MAILbusアーキテクチャを実装したDECの製品名．メッセージルータはX.400メッセージ転送エージェントと類似のものである．

メディア（通信媒体；medium）
ネットワークで用いられる同軸ケーブルのようなケーブル．英語では霊媒のこともこういう．

メディアインターフェイスコネクタ(Media Interface Connector: MIC)
光ファイバをFDDIコントローラに接続するコネクタ．

メンテナンス操作プロトコル (Maintenance Operation Protocol: MOP)
ネットワークを介した遠隔ブートのときや遠隔の計算機にコンソールを接続する場合に用いる，DECnetの特殊用途プロトコル．

モデム(modem)
変調/復調器(modulator/demodulator)を略したもの．計算機からディジタルのデータを受け取り，それを電話線で送れるようにアナログ信号に変調する装置．計算機をアナログ広帯域システムに接続するのにも用いられる．

モード (mode)
外部変数モードの項を参照．

モニタビット(monitor bit)
トークンリングの概念．0よりも大きな優先度をもつフレームがいつまでもリングを回り続けることがないように，活性モニタによってモニタビットが反転される．

有限状態機械(finite state machine)
入力と内部状態に基づいて他の状態への遷移および出力を行う一連の状態としてネットワークモジュールを記述する方法．

ユーザデータグラムプロトコル（User Datagram Protocol）
TCP/IPプロトコル階層の一部．UDPはトランスポート層の役割を果たすが，TCPとは異なり，データの配達を保証しない．

ユーザ認証方式（User Authentication Method）
AppleTalkにおける概念で，資源にアクセスする前に，ファイルサーバに対してユーザを識別する方式を表す．

要求(request)
マネージャからエージェントへのネットワーク管理呼出しであり，通常は特定の属性値を収集したり送り返したりするためのものである．[SunNet Manager]

要求ビットマップ(request bitmap)
AppleTalkトランザクションプロトコルのフィールドで，返答のどの部分が送られたり受け取られたりしたかを記録しておく．

要素(element)
管理の対象となる個々のオブジェクト．例えば，「mgrhostと呼ぶマシン」，「イーサネットインターフェイスie0」，「ビルディング14」など．[SunNet Manager]

要素型(element type)
特定の種類の要素の構造定義．例えば，component.sun3やbus.ethernetなど．[SunNet Manager]

要素カテゴリー(element category)
要素型の組．SunNet Manager Consoleは，バス，構成要素，コネクションおよびビュー(view)という4つの要素カテゴリーを定義する．[SunNet Manager]

呼出し識別子(call descriptor)
Netwiseの定義したデータ型 call_desc をもつデータ構造．特定のRPC要求に関する情報を保持しており，呼び出される側で用いられる．[Netwise RPC Tool]

呼出し方式(calling style)
RPCにおける，呼出しの操作的な性質．oneshot, separable, idempotent, batched, broadcast, no-replyなどがある．[Netwise RPC Tool]

予約フィールド (reservation field)
トークンリングのパケットのフィールドで，ノードがアクティブモニタに対しある送信優先度のデータをもっていることを知らせる．

撚り対線(twisted pair)
電話を端子板に接続するのに用いられるような一対の(あるいは何対かの)線．撚り対線はまたイーサネットやトークンリング, その他のデータリンクの物理伝送媒体としても用いられる*．(*訳注：米国では電話の配線に撚り対線が通常用いられるが，日本では(撚っていない)平行2芯ケーブルが通常用いられる．)

ランデブー(rendezvous)
エージェントがデータやイベントの報告を送るプロセス．rendezvous processとしても知られる．[SunNet Manager]

リトルエンディアン(little endian)
最小のアドレスのオクテットが最下位であるような複数オクテットからなるデータ．

リピータ(repeater)
2つまたはそれ以上のケーブルセグメントを接続するイーサネット装置．リピータは一方のセグメントから受け取った信号のタイミングの調整，増幅を行ってから，他のすべてのセグメントに送り出す．

粒度(granuarity)
オペレーティングシステムのロックマネージャで用いられる用語．ロックマネージャがファイル全体をロックする場合，これは粗い粒度をもつロックとなる．ロックマネージャが単一のレコードをロックする場合，これは細かな粒度となる．粒度はDBMSのような特定のアプリケーションの性能に影響する要因の1つである．

領域 (area)
ルーティング層で使われる用語．レベル1のルータは1つの領域の中のルーティングを行う．レベル2のルータは領域間のルーティングを行う．DECnetの場合は1つの領域の中に1023個までのノードが収容でき，また63個までの領域が収容できる．OSPFも同様の概念を用いている．

ルータ(Router)
ネットワーク上のトラフィックを経路づけするための専用ハードウェア．代わりにSunやVAXのような汎用システムの一部を利用することもできる．

ルーティングディレクトリ(routing directory)
ネットワーク層により管理されているデータベースで，特定のネットワークへ着くためにはどの経路を通ればよいかを決定するのに用いられる．

ルーティングテーブル (routing table)
ネットワーク層で管理されるディレクトリで，インターネットワーク上のノードのアドレスとそこへの到達経路が格納されている．

ルーティングドメイン(Routing Domain)
同じルーチング手続きに従って動作し，1つの管理ドメイン内に完全に含まれる，一連のエンドシステムおよび中間システム．[RFC 1136]

ルート(root)
Unixのスーパーユーザ．Unixシステムにおいてアクセス特権をもつただ1つのアカウント．

ルートディレクトリ(root directory)
名前空間やファイルシステムの頂上．

歴史的プロトコル(Historic Protocol)
TCP/IP標準における区分．決してInternetにおける標準とはならないであろうプロトコルのこと．これはその後の発展や，興味をもたれていないことなどによる．[RFC 1140]

ロータリ(rotary)
ロータリの例は，ダイヤルアウトサービスにおいていくつかの回線が利用可能である場合である．ロータリサービスでは，ユーザに各々の回線を指名させるのではなく，利用可能な最初の回線にユーザを接続する．ロータリはこのように1つのアドレスを用いてすべてにアクセス可能な複数のインスタンスである．サービス供給者はそのアドレスへのすべての要求を捕獲して，特定のアドレスと分配する．

ロック (locking)
あるデータに他のユーザがアクセスできないようにすること．

ロック管理 (lock manager)
オペレーティングシステムの一部で，同じデータへの複数の要求がデータの一貫性を破壊しないことを保証する．

論理的 (logical)
物理的な詳細に触れないこと．例えば，あるデータを論理的に要求するということは，データがどこに存在するか，どのようにそれを得るかを知る必要のないことを意味する．

論理リンク制御 (Logical Link Control)
IEEE 802.2標準で定義されている，データリン

ク層の上位部分．Logical Link Control層は，データリンクサービスのユーザ(多くの場合はネットワーク層である)に均一なインターフェイスを与える．データリンク層のLLC副層の下には，媒体アクセス制御副層(MAC)がある．MAC副層は，LLCからのパケットを受け取り，特定のデータリンク(イーサネットやトークンリング)に送り出す役割をもつ．

ワークグループ(workgroup)

いっしょに仕事をする人達のことを指す流行語．いくつかのコンピュータがワークグループネットワークとして知られる小さなネットワークに孤立されることがある．そこで何かが達成されるかどうかは別の問題である．

割当て(allocation)

トランスポート層プロトコルで使用される概念．割当てとは，一度に許される未確認トラフィックの量である．

割当て番号 (assigned number)

Internetの標準の一部として公式に割り当てられた番号．

割出し(trap)

特定の条件(通常は誤り)が発生したときに実行される一連のコードを表すプログラミング上の概念．

索引

訳者略歴

相田　仁
1985年 東京大学大学院博士課程(電気工学)修了
現職　東京大学助教授／工学部電子工学科

小池汎平
1990年 東京大学大学院博士課程(情報工学)修了
現職　東京大学講師／工学部電気工学科

Sunネットワーク詳説

平成5年9月30日　発　行

訳　者　相　田　　　仁
　　　　小　池　汎　平

発行者　鈴　木　信　夫
　　　　出版事業部　深山恒雄

発行所　丸善株式会社　東京・日本橋

出版事業部　〒113　東京都文京区本郷二丁目38番3号
編集部　電話 (03) 5684 - 5081／(03) 5684 - 2458
営業部　電話 (03) 5684 - 5571／(03) 5684 - 2456
郵便振替口座　東京 7 - 5 番

印刷・富士美術印刷株式会社／製本・株式会社 松岳社

ISBN 4 - 621 - 03888 - 5 C3055　　Printed in Japan

主要目次

装釘　横山晴夫

定価＝7,519円（本体7,300円・税219円） ISBN4-621-03888-5 C3055 P7519E

